21021
23

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLÁN"

PROBLEMA DE FLUJO A COSTO MÍNIMO MÉTODO DE TRANSICIÓN DE ESTADO

TESINA

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACÍON

PRESENTA:

KONDRATENKO BEGER ALEXANDRA

ASESOR:

LIC. GUADALUPE DEL CARMEN RODRIGUEZ MORENO

JULIO 2003

A

CONTENIDO

CONTENIDO

"Los que mandan, generalmente mueven las manos
y dicen:"He considerado todas las alternativas",
pero eso es casi siempre basura.
Lo más probable es que no pudiesen estudiar
todas las combinaciones".[1]

George B. Dantzig

[1] Entrevista publicada en *The College Mathematical Journal*, marzo de 1986.

INTRODUCCIÓN

La Optimización en Redes estudia problemas de optimización que se pueden estructurar en la forma de una red, es decir, de un grafo. Esta área de estudio enlaza las dos grandes ramas de la optimización, que son la continua y la discreta; y para la construcción de algoritmos aplican dos enfoques: los métodos exactos y los métodos aproximados.

El enlace entre estas dos áreas es construido en optimización de redes mediante la Teoría de Grafos y los criterios para la selección de algoritmos provienen de la teoría de la complejidad.

Entre los problemas que pueden ser abordados se encuentra el diseño y análisis de grandes sistemas, tales como redes de transporte, redes de potencia y sistemas distribuidos de cómputo y de comunicación. Las primeras dos pertenecen a la Investigación de Operaciones, surgiendo los problemas clásicos de transporte y transbordo.

El problema de transbordo con capacidades es el modelo más general del problema de flujo con costo mínimo, el cual incluye al problema de transporte con o sin capacidades y el problema de asignación. Estos modelos se usan en un gran número de aplicaciones, como el transporte de bienes, el diseño de redes de comunicación y de tuberías, la asignación de hombres a trabajos, y la planeación de producción.

El problema de transbordo con capacidades son problemas de flujo a costo mínimo donde el objetivo es determinar cómo un bien debe fluir a través de los arcos de una red para minimizar el costo de transbordo.

El problema de transporte fue propuesto por Hitchock en 1941 *ref.(1)* y por Koopmans en 1946 *ref. (2)*, y se busca determinar a qué precio un bien debe de ser transportado de un lugar a otro a través de una red para minimizar el costo de transporte. Además, este problema se puede formular como un problema de programación lineal. Ambos autores presentaron métodos computacionales que ahora serían llamados simplex primal, es decir, a partir de una solución factible primal inicial resuelven un problema de programación lineal. En 1951 Dantzig *ref. (3)* presentó su algoritmo simplex, y desarrolló una variante especial para el problema de transporte del algoritmo simplex.

Existen varios algoritmos de tipo primal, dual y primal-dual que resuelven este tipo de problemas. La opinión sobre cuál método es mejor, más fácil o más eficiente está muy dividida. Los autores como Hitchcock, Koomans, Dantzig, Brown y Bradley prefieren los algoritmos primales. Sin embargo, los autores, como Balas y Hammer, Armstrond, Glover creen que los métodos duales son más eficientes que los métodos primales. Pero también se reconoce que los algoritmos primales-duales son muy eficientes y son recomendados por autores como es Hatch. Dependiendo del tipo

del problema que se tenga, de su complejidad y algunos otros factores se determinará qué modelo de programación lineal debe utilizarse para resolver el problema.

En este trabajo se da a conocer el algoritmo de transición de estado, un algoritmo de tipo primal-dual para el problema de flujo a costo mínimo que consiste en encontrar cómo deben fluir los bienes a través de una red, de tal manera que se cumpla la ley de conservación (todo lo que entra debe de ser igual al que sale) a costo mínimo[2]. Este problema contiene casos particulares como por ejemplo: problema de flujo máximo, de ruta más corta, de transporte, de transbordo y de asignación.

El contenido de este trabajo de investigación requiere un conocimiento básico sobre la teoría de gráficas, de programación lineal y de álgebra lineal. Sin embargo, en el capítulo uno se presentan los conceptos necesarios para una mejor comprensión del trabajo y en el apéndice está la información para recordar los conceptos básicos de programación lineal y del método simplex.

Además, se introduce la notación y terminología básica de teoría de gráficas y del problema de flujo, así como algunos resultados de esta área y teoremas que servirán para desarrollar los capítulos siguientes.

En el capítulo dos se verán algunos de los problemas principales de redes y cómo pueden ser transformados en problemas de flujo a costo mínimo, y así aplicar el método de transición de estado, el cual se desarrolla y se implementa en el mismo capítulo. También se revisará uno de los resultados más importantes para el método de transición de estado: el teorema de circulación entera, y también se verán algunas definiciones y las características de las matrices totalmente unimodulares.

En el capítulo tres se describe el moderno enfoque del análisis de la complejidad de los algoritmos, principalmente los algoritmos con cotas polinomiales, la técnica de escalamiento de Edmonds y Karp, y cómo se aplica al método de transición de estado. Y finalmente, se comentarán y se compararán algunos de los algoritmos más importantes que resuelven el problema de flujo a costo mínimo.

En el capítulo cuatro se describe cómo podemos aplicar el método de transición de estado, para resolver el problema de planeación de proyectos, problema de transporte y transbordo y el problema de asignación.

[2] El método de transición de estado, presentado en el trabajo, se conoce en inglés como Out-of-Kilter.

CAPÍTULO I

CONCEPTOS DE TEORÍA DE GRÁFICAS Y DEL PROBLEMA DE FLUJO A COSTO MÍNIMO

En este capítulo se introduce la notación y terminología básica de teoría de gráficas y del problema de flujo, así como algunos resultados de esta área que servirán para desarrollar los capítulos posteriores.

1.1 ELEMENTOS BÁSICOS DE LA TEORÍA DE GRÁFICAS

Las gráficas son importantes para modelar muchas de los problemas de Investigación de Operaciones, principalmente en programación lineal, así como situaciones de la vida diaria, relaciones interpersonales, circulación vial, rutas de aviación, y un sin fin de situaciones. Y las redes, que se usarán para el desarrollo de este trabajo, se representan a través de las gráficas. Por eso, antes que nada, vamos a definir qué es una gráfica, cuáles son sus elementos, qué relación tienen entre si, y los tipos de las mismas.

Definición. – Una *gráfica* es un conjunto V finito, no vacío, cuyos elementos son llamados vértices o nodos y un conjunto A de pares ordenados de puntos de V, i.e., $A \subseteq V \times V$, cuyos elementos son llamados arcos o líneas de G. La gráfica se denota por $G = (V, A)$.

Se dice que el arco (i, j) une a los vértices i y j. Para un arco (i, j), el vértice *i* es llamado *punto inicial o extremo inicial*, y el vértice *j punto final o extremo final* del arco.

Los arcos que unen a los nodos de una gráfica pueden ser de dos tipos: líneas que tienen cierta dirección o las líneas sin el sentido de dirección. Ésta característica divide las gráficas en las dirigidas y no dirigidas.

Definición. – Una *gráfica dirigida* G(V, A) es en la que el sentido de los arcos es de suma importancia. Las *gráficas no dirigidas*, son aquellas en las cuales no nos importa la dirección de los arcos.

Podemos tener una representación geométrica de una gráfica dibujando los vértices como puntos y los arcos como líneas que unen pares de puntos. En la figura 1.1.1 se muestran ejemplos de las gráficas dirigidas:

Fig 1.1.1

A veces, en el análisis del problema, lo que interesa es solamente ver el comportamiento de una parte de la gráfica original, o de un arco en particular, por lo tanto, es más practico trabajar sobre la subgráfica, que es un modelo reducido que conserva características y los datos originales.

Otro caso sería que nos interesara poder dividir el problema original en dos o más sin afectar el proceso general.

Definición. – Una gráfica g es una ***subgráfica*** de G, si todas las líneas y vértices de g están contenidos en G, y además, cada línea de g tiene los mismos vértices terminales que en G.

En la figura 1.1.2 se presenta el ejemplo de la gráfica no dirigida G y algunos de sus subgráficas posibles:

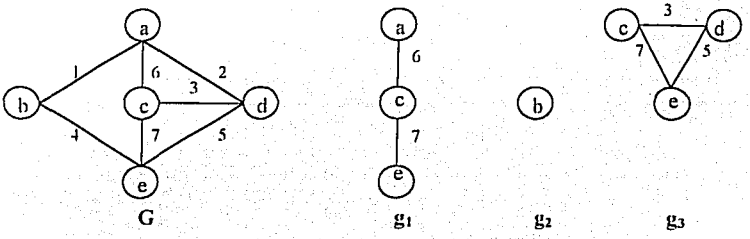


Fig 1.2

La figura 1.1.3 le muestra ejemplo de las subgráficas de una gráfica dirigida G:

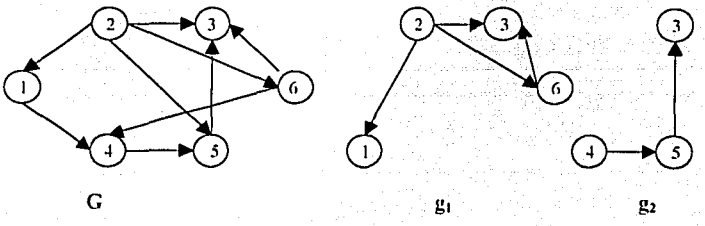


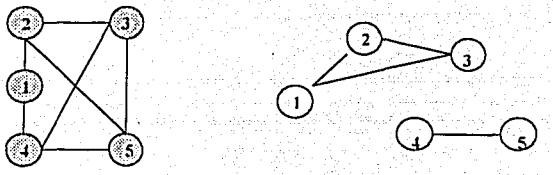
Fig 1.1.3

Entre las definiciones anterioriores vimos qué es una gráfica dirigida. Ahora, veremos la definición de la gráfica dirigida cuando ésta contiene entre todo par de nodos los arcos de doble sentido.

Definición. – Una ***gráfica simétrica*** es aquella, en la cual para cada línea (v_i, v_j) existe otra línea (v_j, v_i), como se puede ver en la figura 1.1.4.

Fig 1.1.4

Definición.- Una ***gráfica*** es ***conexa o conectada*** si entre cualquier par de vértices existe al menos una ruta. En caso contrario – es una gráfica desconectada. Los ejemplos de ámbos se muestran en la siguiente figura 1.1.5.



Si *No*

Fig 1.1.5

A continuación se verá la relación que se tiene entre los elementos de una gráfica. i.e., entre un nodo y el arco; entre dos líneas y entre dos vértices.

Definición. - El ***vértice*** j se llama ***sucesor*** del vértice i si existe un arco. (i, j). con vértice inicial *i* y vértice final *j*. Igualmente. el ***vértice*** j se llama ***predecesor*** del vértice i si existe un arco de la forma (j. i).

Definición. – Dos ***arcos*** son ***adyacentes*** si tienen al menos un vértice en común. Si el vértice *i* es el punto inicial de un arco el cual no es un lazo, entonces, el arco y el vértice i son *mutuamente incidentes.*

Por lo tanto podemos decir que si dos personas se conocen, entonces estas personas son adyacentes una a la otra. Si una calle pasa por una glorieta, entonces la glorieta y la calle son mutuamente incidentes. Todos los vuelos que llegan a un mismo aeropuerto son vuelos adyacentes.

Ahora veremos qué es una ruta, una cadena, una trayectoria y la diferencia entre ellas.

Definición.- Sea (1, 2, ..., n) una secuencia de vértices distintos con la propiedad de que (i, i+1) ó (i+1, i) es un arco para i = (1,..., n-1); si para cada i una de estas dos posibilidades se cumple, la secuencia resultante de vértices y arcos se denomina una ***ruta o paseo*** del vértice 1 al n. En otras palabras, es una secuencia finita de vértices y líneas empezando y terminando en vértices, donde cada línea es incidente con los vértices anteriores y posteriores.

Definición. - Una ***cadena*** entre los nodos i y j es una secuencia de líneas (arcos) que conecta a estos dos nodos. Cuando también se especifica la dirección en que se recorre la cadena, se le dará el nombre de ***trayectoria***.

Una ***cadena*** es ***simple*** si no contiene ciclos. Y una trayectoria o un camino es ***paseo abierto*** si no se repiten los vértices.

Como podemos notar una ruta difiere de una cadena ya que permite la posibilidad de tener arcos con dirección opuesta a la orientación del vértice 1 al n.

Las nociones de ciclo y enlace que se verán en las definiciones siguientes son muy importantes para el desarrollo del método de transición de estado.

Definición. Un ***circuito*** o ***ciclo*** es una ruta cerrada, donde el único vértice que se repite es el primero al final. O sea, un ciclo es una cadena que empieza y termina en el mismo nodo. Una gráfica es acíclica si no tiene ciclos.

En el siguiente ejemplo de la figura 1.1.6 vamos a identificar algunas de las rutas abiertas y cerradas. Supongamos que se tiene la siguiente gráfica:

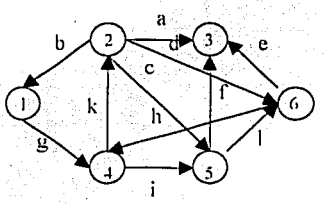


Fig 1.1.6

Entonces, la secuencia de nodos y vértices propuestas se clasificarían como:

a) (1, g, 4, i, 5, f, 3) – es una ruta abierta;
b) (2, c, 5, l, 6, e, 3) – una ruta abierta;

c) (2, b, 1, g, 4, k, 2) – es un circuito;
d) (4, i, 5, l, 6, h, 4) – un circuito.

Definición.- Sea [S, T] = {(u,v)∈ A | u ∈ S, v ∈ T}. Un *conjunto de corte* es un subconjunto C ⊆ A de la forma C = [S, T], donde ∅ ≠ S ⊆ V y T = V\S.

Definición.- Un *enlace* es un conjunto de corte mínimo, es decir, un enlace es un conjunto de corte al cual no se le puede quitar ningún arco, de tal forma que el conjunto resultante siga siendo un conjunto de corte.

Hay que hacer la observación de que un conjunto de corte efectivamente corta o separa la gráfica. Si quitamos el conjunto de corte C, la gráfica queda separada en dos partes: la de nodos S y la de nodos T.

Veamos el siguiente ejemplo, donde S = [s, 1, 2, 3], T =[4, 5, t] y C = [(1,5), (2,5), (2,4), (3,4), (3,t)].

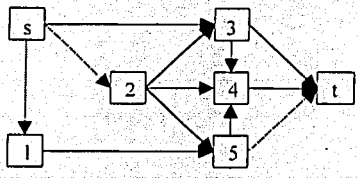


Fig 1.1.7

Con este conjunto de cortes C la gráfica queda separada como se puede apreciar en la figura 1.1.8:

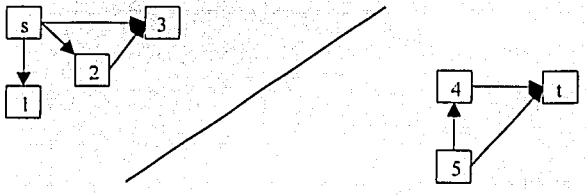


Fig 1.1.8

Como ninguno de los arcos que forman el conjunto de corte puede ser retirado del conjunto sin que deje de ser un conjunto de corte, el conjunto de corte es un enlace.

Ahora se verá otro ejemplo que no es un enlace.

Ejemplo.- S = [s], T = [1, 2, t], y C = [(s,1), (s,2), (s, t)].

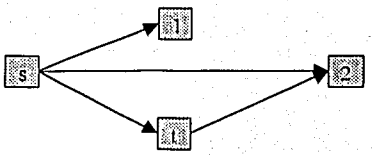


Fig 1.1.9

Con este conjunto de corte la gráfica queda separada en dos, como se muestra en la figura 1.1.10:

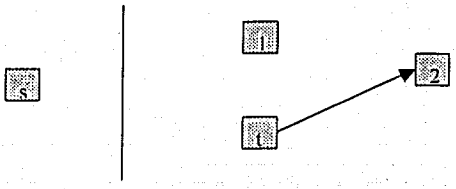


Fig 1.1.10

Pero no es mínimo, ya que si le quitamos al conjunto de corte el arco (s,1), el conjunto de corte sigue siendo conjunto de corte y la gráfica queda separada de la siguiente forma (fig 1.1.11):

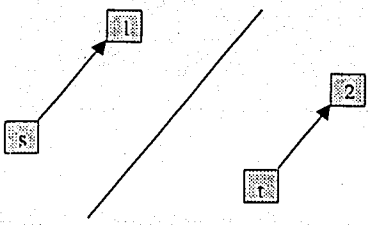


Fig 1.1.11

Por lo tanto, no es un enlace.

Proposición.- Cualquier conjunto de corte C es la unión de enlaces ajenos.

Demostración. - Haremos la demostración por inducción sobre k, el número de elementos del conjunto C.

Para k = 1 el conjunto C sólo consta de un elemento, y este único arco es un conjunto de corte; y es mínimo por que sólo tiene un elemento, por lo tanto es un enlace.

Supongamos que la hipótesis de inducción se cumple para k = n, i.e., cualquier conjunto de corte con n arcos es la unión de enlaces ajenos.

Para k = n+1 tenemos un conjunto de corte con n+1 arcos. Este conjunto de corte puede ser un enlace por si mismo, en cuyo caso ya tenemos la demostración, o puede no ser un enlace. Si no es un enlace le quitamos un arco, de manera que siga siendo un conjunto de corte, y le aplicamos la hipótesis de inducción. Y tenemos un conjunto de corte con n+1 arcos que son la unión de enlaces disjuntos.

El siguiente teorema que fue desarrollado en 1961 por Minty *ref.(4)*, se utilizará en el capitulo dos en la explicación del método de transición de estado. Además el programa de implementación computacional del método de transición de estado presentado en este trabajo se basa en el teorema de coloración.

Teorema.- Teorema de Coloración

Sea G una gráfica dirigida con un arco (s, t) dirigido. Entonces, para cualquier coloración de los arcos verde, amarillo y rojo con (s, t) pintado de amarillo, exactamente una de las siguientes condiciones se cumple:

1.- El arco (s, t) esta contenido en un ciclo de arcos amarillos y verdes, en el cual todos los arcos amarillos tienen la misma dirección.

2.- El arco (s, t) esta contenido en un enlace de arcos amarillos y rojos, en el cual todos los arcos amarillos tienen la misma dirección.

Demostración.- Veamos la gráfica como en una red de calles, donde los arcos verdes son las calles de doble sentido, los arcos amarillos son las calles de un solo sentido, y los arcos rojos son las calles cerradas al trafico. Empezando en la intersección de una calle representada por un vértice

t, o es posible para el trafico moverse de t a s, o no es posible. Si existe algún camino, entonces existe una ruta – (t, s) mínima de arcos amarillos y verdes, con todos los arcos amarillos dirigidos de t a s. Esta ruta junto con el arco (s, t) forman un ciclo que satisface la condición 1. (fig 1.1.12)

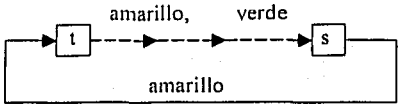


Fig 1.1.12

Si no hay ningún camino para que el trafico llegue de t a s, entonces un enlace que satisface la condición 2 y se construye de la siguiente manera: Sea T el conjunto de todos los nodos accesibles al trafico desde t y sea S el conjunto complementario. No puede haber arcos amarillos dirigidos de T a S ni arcos verdes entre S y T en ninguna dirección. De otra forma, uno o más de los vértices en S estarán accesibles al trafico de T, contrario a lo asumido anteriormente. De aquí se sigue que todos los arcos entre S y T deben ser arcos rojos en cualquier dirección, o arcos amarillos incluyendo al arco (s, t) dirigidos de S a T. Por la proposición 1 el conjunto de corte (S, T) contiene un enlace que satisface la condición 2. Como se puede observar en la gráfica de la figura 1.1.13.

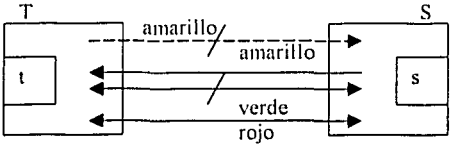


Fig 1.1.13

A continuación veremos algunas de las matrices con una gráfica G (V, A); donde la cardinalidad de V es n y la de A es m.

1.2 GRÁFICAS DIRIGIDAS

Definición.- Matriz de incidencia $M = [m_{ij}]$, es de $n \times m$, donde $m_{ij} = 1$ si el arco (i, j) sale del nodo i, $m_{ij} = -1$ si el arco (i, j) llega al nodo j, y $m_{ij} = 0$ de otra forma, o sea:

$$m_{ij} = \begin{cases} 1 & \text{si el arco } a_j \text{ sale del vértice } v_i \\ -1 & \text{si el arco } a_j \text{ llega al vértice } v_i \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Donde los renglones son los vértices y las líneas – columnas.

Ejemplo.- Se tiene la siguiente gráfica G, (fig 1.2.1):

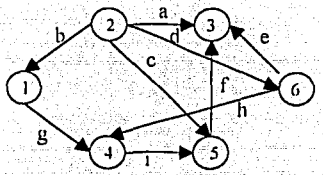


Fig 1.2.1

La matriz M(G) correspondiente será:

$$M(G) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} a & b & c & d & e & f & g & h & i \end{array} \\ \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Hay que notar que las matrices tanto de incidencia, como de adyacencia pará las gráficas dirigidas y no dirigidas, tienen ciertas propiedades. Vamos a mencionar únicamente las que tienen importancia para el desarrollo de esta tesina.

Las propiedades de la matriz de incidencia de una gráfica dirigida:

1) En cada columna existe exactamente un 1 y un −1, excepto en caso de un buele, en el cual existirá un ±1, (pero no trataremos en este trabajo este tipo de gráficas).
2) Las líneas paralelas producen columnas iguales.
3) Si se intercambian renglones y columnas con sus etiquetas respectivas, la matriz sigue representando la misma gráfica.
4) La suma por columnas siempre es cero.
5) Esta matriz está definida para graficas dirigidas sin lazos.

Definición.- Matriz de adyacencia $A = [a_{ij}]$, es la matriz cuadrada de n×n, en la cual:

$$a_{ij} = \begin{cases} 1 & \text{si existe un arco que sale del } v_i \text{ y llega al vértice } v_j \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Ejemplo.- La matriz A de la gráfica anterior, (fig 1.14) será:

$$A(G) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Las propiedades de la matriz A(G):

1) A(G) es cuadrada.
2) El diagonal principal tiene sólo ceros. (en caso de que tuviera un uno, este representaría un buele, pero no se trataran las gráficas asi en esta tesina).
3) Un renglón de ceros corresponde a un vértice final.
4) Si A(G) es simétrica, entonces G es simétrica.
5) Si en la gráfica no existen las líneas paralelas, entonces, el número de unos es igual al número de líneas de la gráfica.

En este trabajo no se van a tratarse los siguientes casos, pero es importants conocerlos para el análisis:

6) Una columna de ceros representa a un vértice aislado.
7) Si se tiene un renglón y una columna de ceros, entonces se trata de un vértice aislado.

La matriz de adyacencia guarda las características de la gráfica, y se utiliza para almacenarla en la computadora.

1.3 DEFINICIONES DE PROGRAMACIÓN LINEAL

Toda disciplina científica emerge de la convergencia de un interes en alguna clase de problemas y del desarrollo de métodos, técnicas e instrumentos científicos adecuados para resolver estos problemas. Los fundamentos matemáticos de los modelos lineales discretos se basan en la teoría de las desigualdades lineales. Y como antecedente de Programación lineal se encuentra el planteamiento del problema de transporte. Las problemas de este género las podemos representar gráficamente en la forma de una red. Ejemplos de algunos problemas que pertenecen a programación lineal se verán en el capitulo II.

Un problema de programación lineal es un problema de minimizar o maximizar una función lineal que contiene restricciones lineales del tipo de igualdad, desigualdad o las dos. La forma general del modelo es el siguiente:

$$\text{Max } z = \sum_{j=1}^{n} c_j x_j \qquad \text{función objetivo}$$

$$\text{sujeto a: } \sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1, 2, \ldots, m \qquad \text{las restricciones}$$

$$x_j \ge 0, \; j = 1, \ldots n. \qquad \text{var. de decisión}$$

La función que se está maximizando o minimizando se llama *función objetivo*. Y las cantidades desconocidas que se deben de determinarse en la solución del modelo matemático del problema se llaman *variables de decisión*, y por lo general se representan por:

$$x_1, x_2, \ldots, y_1, y_2, \ldots, \text{etc.}$$

Al construir el modelo del problema real para incluir las limitaciones que ocurran se usan las *restricciones*. Son las que limitan las variables existentes a valores factibles y se representan de siguiente manera:

$$a_1x_1 + a_2x_2 \le b$$

Todas las variables que se hacen igual a 0 se llaman ***variables no básicas*** y los variables restantes son las ***variables básicas***. Y al obtener la solución de dicho modelo, éste se llamará – ***solución básica.***

Definición - La solución será ***básica factible*** si todas las variables básicas son no negativas, i.e, es una solución básica en la que todas las m variables básicas sean ≥ 0.

Pero cuál es la solución factible. Enseguida se define la dicha solución factible y la solución óptima a un problema de programación lineal.

Definición - Una ***solución factible*** es una solución para la cual se satisfacen todas las restricciones. Y una ***solución óptima*** es una solución factible que tiene el valor más favorable de la función objetivo, i.e., el valor menor o mayor, dependiendo de si el objetivo es maximizar o minimizar. En otras palabras, si todas las $u_i - u_j \geq 0$ para todas las variables no básicas (en caso de Max) y son ≤ 0, en caso de Min $u_i - u_j$.

Ahora veremos los conceptos básicos de programación lineal de redes de flujo.

Definición.- Una *red* es una gráfica conectada cuyos arcos o líneas pueden considerarse como ductos, a través de los cuales pasa la información. Toda red contiene un origen y un destino.

Hay que notar que, una *red conexa* es aquella en donde existe por lo menos una cadena que conecta a cada nodo con el resto de los nodos de la red. Una *red inconexa* es aquella que no esta conectada.

Ahora definiremos una red de transporte que es la que nos interesa para el desarrollo de los capítulos posteriores.

Definición.- Sea G = (V, A) un grafo de orden n. Diremos que es una red si se cumple:

- G es conexo y no tiene bucles.
- Hay definida una función numérica sobre los arcos del grafo, que notaremos C, y la cual llamaremos capacidad que asigna a cada arco del grafo un valor no negativo.

Además, si

- Existe un vértice y sólo uno, tal que a el no llega ningún arco: La entrada a la red, nodo fuente.
- Existe un vértice, y sólo uno, tal que de el no sale ningún arco: La salida de la red, nodo sumidero o final.

Entonces tenemos una *red de transporte.*

Por medio de una red de transporte vamos a "pasar" o mover algún bien de un punto hacia el otro. Y la cantidad de este "bien", como lo veremos más adelante es llamado flujo. O, dicho de otra forma, podemos representar en una red R y su gráfica dirigida G = (V,A) conexa y sin lazos, junto con una función c: $A \rightarrow [0, +\infty]$, llamada la *función capacidad* de R. Si el arco (i, j) ∈ A el numero

c_{ij} es llamado la capacidad del arco (i, j). Esta capacidad puede pensarse intuitivamente como la cantidad máxima de algún bien que puede "fluir" a través del arco por unidad de tiempo.

Definición.- Sean s y t dos vértices distintos de R. Un *flujo* de s a t con valor v es una función x: $A \rightarrow [0, \infty)$ tal que satisface:

$$0 \leq x_{ij} \leq c_{ij} \quad \forall (i,j) \in A \qquad (1)$$

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} v & \text{si } i = s \\ 0 & \text{si } i \neq s, t \\ -v & \text{si } i = t \end{cases} \qquad (2)$$

Si (i, j) $\in$ A, x_{ij} es la cantidad de flujo que va en la dirección de i a j, la desigualdad (1), simplemente indica que en cada arco la cantidad de flujo que pasa no debe exceder la capacidad del arco. La ecuación (2) es la ley de conservación, que establece que en cada vértice i distinto de s ó t, lo que sale del arco

$$\sum_j x_{ij}$$

debe ser igual a lo que entra en él:

$$\sum_j x_{ji}$$

El nodo *s* es llamado la *fuente*, y *t* es el *destino*. Los vértices restantes se conocen como los vértices *intermedios*. (si los numeramos, el vértice *s* tendrá el número 1 y *t* el n). Si se piensa en una red, como la de transporte, el nodo fuente puede ser un centro de producción, mientras que el vértice destino correspondería a un mercado. También puede pensarse en una red, como la de dipolos por la cual fluye corriente eléctrica continua; (en este contexto, la ley de conservación es la primera ley de Kirchhoff), del tráfico urbano, de transporte colectivo, de comunicaciones, de inventarios, de presas, de flujo de dinero, de la asignación de recursos entre otros.

Un flujo en una red se puede representar geométricamente dibujando la gráfica dirigida colocando sobre cada arco: la cantidad de flujo y la capacidad del arco, como se ve en la figura 1.3.1:

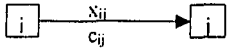


Fig 1.3.1

Enseguida veremos tipos de flujo que existen en una red.

Definición. – Un *flujo factible* es un flujo que satisface todas las restricciones del problema. Y si además el flujo factible es una solución básica de las restricciones del problema de redes, entonces este se denomina como *flujo factible básico*.

En muchos problemas prácticos, las variables de decisión alcanzan un sentido sólo si tienen valores enteros como en el caso de asignación de hombres, máquinas o vehículos a las actividades, entonces el flujo en la red asociada será en cantidades enteras.

Definición. – Un ***flujo factible*** $x = (x_{ij})$ es ***entero*** si para cada arco (i, j) el flujo x_{ij} es cero o un número entero positivo.

Un ***flujo*** es ***completo*** si todo camino que va de la fuente al sumidero contiene al menos un arco saturado *ref.(5).*

Cuando estamos buscando las opciones de llegar de un nodo a otro en una gráfica, los arcos a traves de los cuales nos estamos moviendo los llamaremos arcos de avance.

Definición.- Los arcos (i, i+1) que pertenecen a la ruta son ***arcos de avance*** de la ruta, los demás arcos son de retroceso.

Para obtener la solución, sea ésta la óptima o no a un problema en el proceso del desarrollo del método de transición de estado vamos ir manipulando los datos asociados a cada arco. En el caso del flujo, si el flujo de un arco puede ser aumentado y dicho arco esta contemplado en la ruta, entonces esta ruta tendrá la siguiente definición.

Definición.- P es una ***ruta de flujo aumentante*** con respecto a un flujo dado x si:

$x_{ij} < c_{ij}$ para cada arco de avance (i, j) de P

y

$x_{ij} > 0$ para cada arco de retroceso (i, j) de P

Lo significa que la cantidad del flujo de cada arco debe de ser mayor que cero y menor que la capacidad máxima permitida en dicho arco.

Definición. Sea R = (V, A, x, a) una red, donde x es la función de capacidad y a es la función de costos por unidad de flujo asociadas a sus arcos. Se define como el ***costo de un flujo*** factible a la cantidad

$$\sum_{(i,j) \in A} a_{ij} x_{ij},$$

Donde a_{ij} es el costo unitario del flujo a través del arco (i, j) y x_{ij} es la cantidad de flujo que va de i a j.

Definición. - La ***capacidad de flujo*** de un arco de gráfica dirigida es el limite superior para la magnitud factible (o la cantidad total de flujo) en el arco en dicha dirección. La capacidad de flujo puede ser cualquier cantidad no negativa, incluyendo infinito.

Ahora, sea G = (V, A) una gráfica; la *circulación de G* es una función x: A→ $\mathbb{R}$ tal que:

$$\sum_j x_{ji} - \sum_k x_{ik} = 0 \qquad \forall\ i \text{ (ley de conservación)}$$

Supongamos que también tenemos dos funciones de capacidad

$l: A \rightarrow [0, \infty)$ cota inferior
$c: A \rightarrow [0, \infty)$ cota superior

y se pide que $l_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\ i, j,$

En este caso tenemos un problema de circulación con capacidades. Si además tenemos la función costo:

$$a: A \rightarrow \mathbb{R},$$

Entonces un problema de flujo con restricciones a costo mínimo es:

$$\text{Minimizar} \sum_{i,j \in A} a_{ij} x_{ij}$$

$$\text{sujeto a:} \quad \sum_{i,j \in A} x_{ji} - \sum_{i,j \in A} x_{ij} = 0 \qquad i = 1, 2, \ldots n$$

$$l_{ij} \leq x_{ij} \leq c_{ij} \qquad i,j \in A,$$

Donde el vector de variables de decisión $x = (x_{ij})$ se llama el flujo de la red.

1.4 PROBLEMA PRIMAL

El concepto de dualidad establece que todo problema de programación lineal tiene asociado otro problema de programación lineal conocido como dual. Las relaciones entre el problema original, que se llama primal y el problema dual son muy importantes.

Hay que notar que el problema primal para poder compararlo con el problema dual, debe de estar en su forma estándar, el cual se representa de la siguiente manera:

$$\text{Maximizar } Z = \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeto a:} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i, \qquad i = 1, 2, \ldots, m$$

$$x_j \ge 0, \qquad j = 1, 2, \ldots, n$$

1.5 PROBLEMA DUAL

Al analizar el problema primal y como se obtiene la solución óptima a éste se puede darse cuenta que el problema dual es una forma invertida del primal, para el cual el valor óptimo de $\mathbb{Z}$ en el problema primal es el valor mínimo factible de y_0 en nuevo problema. Por lo tanto el *problema Dual* se representa en forma de:

$$\text{Minimizar } y_0 = \sum_{i=1}^{m} b_i y_i$$

$$\text{sujeto a:} \quad \sum_{i=1}^{m} a_{ij} y_i \ge c_j, \qquad j = 1, 2, \ldots, n$$

$$y_i \ge 0, \qquad i = 1, 2, \ldots m$$

En la tabla 1 están representados dos problema de programación lineal: El Primal y el Dual. *ref. (7)*

			Problema Primal					
			Coeficiente de				Segundo miembro	
			x_1	x_2	.....	x_n		
Problema Dual	Coeficiente de	y_1	a_{11}	a_{12}		a_{1n}	$\leq b_1$	Coeficientes para función objetivo (minimizar)
		y_2	a_{21}	a_{22}		a_{2n}	$\leq b_2$	
		.....	.....				.....	
		y_m	a_{m1}	a_{m2}	.....	a_{mn}	$\leq b_m$	
	Seg. miembro		$\geq$	$\geq$		$\geq$		
			c_1	c_2	.....	c_n		
			Coeficiente para la función objetivo (maximizar)					

Tabla 1

De la tabla, claramente se observa que existe una correspondencia directa entre estas entidades de los dos problema que son: 1) los parámetros para una de las restricciones, son los coeficientes de una variable en el otro problema y 2) los coeficientes de la función objetivo – son los segundos miembros del otro problema.

Ejemplo.- Supongamos que se tiene el siguiente problema primal:

$$\text{Maximizar } Z = 7x_1 + 4x_2$$

$$\text{sujeto a:} \quad x_1 \leq 8$$
$$5x_2 \leq 14$$
$$3x_1 + x_2 \leq 28$$

$$x_i \geq 0, \quad i = 1, 2.$$

El problema dual de este será:

$$\text{Minimizar } y_0 = 8y_1 + 14y_2 + 28\,y_3$$

$$\text{sujeto a:} \quad y_1 + 3y_3 \geq 7$$
$$5y_2 + y_3 \geq 4$$

$$y_i \geq 0, \quad i = 1, 2, 3.$$

Podemos notar que la relación que existe entre estos dos modelos de programación lineal es simétrica i.e., cada solución básica en problema primal tiene una *solución básica complementaria* en el problema dual:

$$(Z =) \sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i \; (= y_0)$$

Si $(x_1, x_2, ..., x_n)$ es una solución factible para el problema primal y $(y_1, y_2, ..., y_m)$ es una solución factible para el dual, entonces:

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i y_i$$

La tercera relación que se observa es: si $(x_1^*, x_2^*, ..., x_{n+m}^*)$ es la solución óptima para problema primal, entonces su solución básica complementaria $(y_1^*, y_2^*, ..., y_m^*, z_1^* - c_1, z_2^* - c_2, ..., z_n^* - c_n)$ debe ser factible y es óptima para el problema dual. Esta relación se resume en el enumerado siguiente:

si $(x_1^*, x_2^*, ..., x_n^*)$ y $(y_1^*, y_2^*, ..., y_m^*)$ son *soluciones óptimas* para los problemas primal y dual, respectivamente, entonces

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$$

Esta última relación se conoce como el *Teorema Dual.*

Hay que notar que si:

1) Uno de los problemas no tiene la solucion factible y el otro tiene una región factible no acotada, permitiendo que el valor de la función objetivo crezca indefinidamente, o

2) Ambos problemas no tienen soluciones factibles,

Esto quiere decir que los problemas primal y/o dual pueden no tener soluciones óptimas.

Ahora veremos conceptos importantes para el método de transición de estado que usaremos para resolver problemas de flujo a costo mínimo en redes, el cual consiste en disminuir las desviaciones que existen entre el flujo actual y el flujo óptimo. Para lograrlo se hace el análisis de cada arco y el ajuste necesario según las reglas que estan explicadas un poco más adelante.

Definición. – Un periodo de tiempo del objeto (no instantáneo) en el cual el objeto esta esperando alguna operación o acción se llama ***estado*** y es una condición durante la vida del mismo.

Como ya se mencionó, en el método de transición de estado, antes que nada hay que definir el tipo de estado en que se encuentra cada arco. Esto se hace según las siguientes reglas:

$$u_i - u_j < 0 \Rightarrow \lambda_{ij} > 0 \Rightarrow x_{ij} = l_{ij} \qquad \forall\, i, j = 1, 2, ..., m$$
$$u_i - u_j > 0 \Rightarrow \gamma_{ij} > 0 \Rightarrow x_{ij} = u_{ij} \qquad \forall\, i, j = 1, 2, ..., m$$
$$u_i - u_j = 0 \Rightarrow l_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\, i, j = 1, 2, ..., m$$

Si éstas se satisfacen, entonces, el arco se encuentra en buen estado y no tendrá ninguna modificación. En caso contrario, el arco primero, tiene que ser ajustado para pasar de mal estado a buen estado y así obtener la solución al problema.

Dicho de otra manera, el estado de los arcos de una red de flujo se define así:

Definición - Un arco se encuentra en ***buen estado***, si y solo si, la solución primal y dual son óptimas. Y el arco (i, j) esta en ***mal estado*** si no cumple con las condiciones de optimalidad.

CAPÍTULO II

MÉTODO DE TRANSICIÓN DE ESTADO E IMPLEMENTACIÓN COMPUTACIONAL

En este capitulo, en la primera parte se verán algunos de los modelos principales de redes y cómo pueden ser transformadas a modelos de flujo a costo mínimo. A estos nuevos modelos se les puede aplicar el algoritmo de transición de estado, que se verá más adelante en el mismo capitulo, para su solución. Además se estudiará uno de los resultados más importantes para el método de transición de estado: el teorema de circulación entera. Y se verán algunas definiciones de las caracteristicas de las matrices totalmente unimodulares. Y en la segunda parte veremos en que consiste el método de transición de estado y el código computacional. Se recomienda revisar el apéndice para recordar los conceptos básicos del método simplex de programación lineal, ya que varios de los conceptos que se verán en este capitulo se basan en estos resultados.

2.1 EL PROBLEMA DE REDES

En la investigación de operaciones existe cierto tipo de problemas de programación lineal con características especiales que se conocen como redes. Las redes tienen una aplicación extensa en diversos campos como son: la planeación, la ingeniería, la administración, la química, la economía, etc. Muchísimas situaciones pueden ser formuladas como los modelos matemáticos de redes. Y son tan importantes y útiles debido a que son de fácil comprensión para personas que tienen poco conocimiento en investigación de operaciones y por su facilidad de representarlos gráficamente. Por su gran campo de aplicación y valiosa ayuda que proporcionan para el entendimiento de los sistemas, ha habido gran actividad en su estudio. Han sido desarrollados muchos algoritmos eficientes para la solución de problemas de programación lineal, inclusive existe más de un algoritmo para el mismo tipo de problema. Los más eficientes algoritmos para los problemas de redes son el método heurístico diseñado por Busacker y Gowen y otro método basado en el teorema de holguras desarrollado por Ford y Fulkerson que se conoce como el método de transición de estado o (out-of-kilter).

El algoritmo heurístico es muy fácil de entender y de usarlo para modelos de tamaño moderado. Al aplicar este método se limita de inmediato el tamaño de las redes que se pueden resolver porque al realizar una nueva iteración se tiene que construir nuevos arcos, los arcos en reversa. Así, si la red original contiene un número significativo de variables, al terminar de resolverla se convierte en un problema con el número de elementos muy elevados y esto resulta ser una inconveniente muy serio, debido a que la memoria de la computadora tiende a saturarse rápidamente.

El algoritmo de transición de estado resuelve el problema dado de flujo en una red a costo mínimo sin modificar la estructura de la red. Es mucho más sencillo y eficiente que el método simplex y sus variantes. Requiere que las capacidades, sean números enteros para todos los arcos y que la red a resolver sea circular. Este algoritmo puede iniciar con cualquier clase de flujo x_{ij}. Si este flujo es factible, entonces el algoritmo lo convierte en óptimo. Si no es así, entonces primero lo convierte en factible y después en óptimo.

2.2 EL PROBLEMA DE FLUJO A COSTO MÍNIMO

Supóngase que se quiere encontrar un flujo factible de valor v, entre el origen s y el destino t, incurriendo en el menor costo posible; al flujo de menor costo se le llama *flujo a costo mínimo*. O sea, el problema de flujo con costo mínimo consiste en:

Dado el valor de un flujo v encontrar la distribución que tenga el costo mínimo, donde el costo de un flujo $x = (x_{ij})$ es:

$$\text{Min } Z = \sum_{i,j} a_{ij} x_{ij},$$

Donde a_{ij} es el costo unitario del flujo a través del arco (i, j) y x_{ij} es la cantidad de flujo que puede pasar por el dicho arco.

Hay que notar que si la cantidad v de flujo requerido es mayor que el valor del flujo máximo el problema no tiene solución, por lo tanto de aquí en adelante supondremos que v es menor que el valor del flujo máximo.

Supongamos que se tiene la siguiente red que requiere el flujo de v = 5 a costo mínimo, (fig 2.2.1):

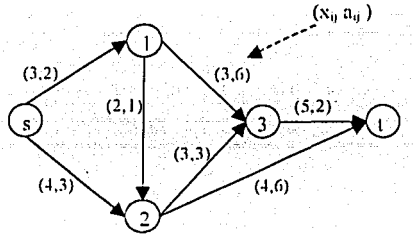


Fig 2.2.1

Resolviendo por alguno de los dos métodos: uno que consiste en determinación de los circuitos negativos en la red marginal, o por el método que se basa en la determinación de las rutas más cortas en esta red que se describen más adelante; se determina un costo de 48 y el flujo de v = 5.

La gráfica muestra la red obtenida del problema original con el flujo de valor 5 y el costo de 48, (los números de los arcos representan el flujo x_{ij}):

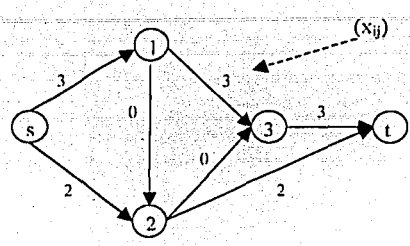


Fig 2.2.2

Analizando la gráfica se observa que si se hacen otras modificaciones en los flujos resultantes de los arcos (1,2) y (2,3) aumentándolos en una unidad y decrementando en uno el arco (1,3) se puede obtener aún menor costo, como se ve en la siguiente figura 2.2.3, igual a 46 con el v = 5:

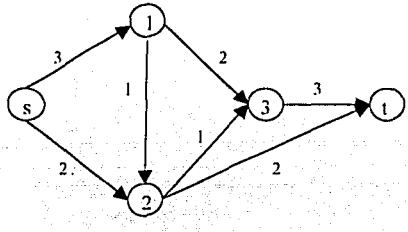


Fig 2.2.3

Por lo tanto aplicando el concepto de *red marginal* se determina si un flujo factible dado es de costo mínimo o no. En caso de que no lo sea, se puede construir otra distribución mejor a partir de la existente.

Definición – Sea R = (V, A, c, a) una red y sea x_{ij} un flujo factible definido en R. La *red marginal* de R, con respecto al flujo x_{ij}, es la red R'(V, $A_1 \cup A_2$, c', a'), donde:

$$A_1 = \{(i, j) \in A \mid x_{ij} < c_{ij}\},$$
$$A_2 = \{(i, j) \mid (j, i) \in A \text{ y } x_{ji} > 0\}$$

c' representa la capacidad de los arcos de R'(x_{ij}) de manera que:

$$c'_{ij} = c_{ij} - x_{ij} \qquad \forall (i, j) \in A_1$$
$$c'_{ij} = x_{ji} \qquad \forall (i, j) \in A_2$$

a' describe el costo unitario del flujo a través de los arcos de $R'(x_{ij})$ de la siguiente forma:

$$a'_{ij} = a_{ij} \qquad \forall (i, j) \in A_1$$
$$a'_{ij} = -a_{ji} \qquad \forall (i, j) \in A_2$$

Como podemos ver, la red marginal indica la manera de obtener otras distribuciones del mismo valor y marca el cambio en el costo al modificar dicho flujo. Al aumentar el flujo a través del arco $(i, j) \in A$ implica incrementar el costo a_{ij} y desminuir el flujo implica decrementar el costo a_{ij}.

Por lo tanto la red marginal $R'(x_{ij})$ del ejemplo anterior con respecto al flujo definido será:
(los números de los arcos de la figura 2.2.4, son la capacidad y el costo del flujo)

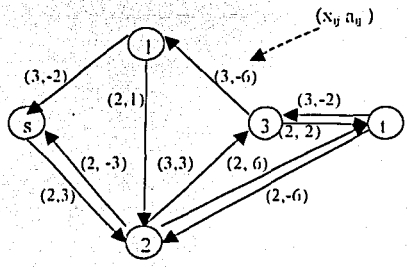


Fig 2.2.4

El siguiente teorema es la forma para determinar el flujo a costo mínimo en una red, *ref. (8)*.[3]

> *Teorema:*
>
> Sea x_{ij} un flujo factible de valor v en una red $R = [V, A, c, a]$ donde x es la función de capacidades y a es la función de costo por la unidad de flujo asociadas a sus arcos.
>
> Entonces el flujo x_{ij} es de costo mínimo si y sólo si no existen circuitos negativos en la red marginal $R'(x_{ij})$.

Otra forma de atacar el problema que no sea suponiendo un flujo inicial cualquiera del valor v requerido, consiste en considerar un flujo factible de valor u ($u < v$) de costo mínimo e ir incrementando el valor del flujo, hasta obtener el valor de v, conservando siempre la optimalidad;

[3] El teorema de flujo a costo mínimo se demuestra en dicha bibliografía.

i.e., que cada flujo que se obtenga sea de costo mínimo. Para lograr esto se usará la red marginal y para incrementar el valor de un flujo dado, habrá que determinar cadenas aumentantes de s a t en la red. Las cadenas aumentantes se explican enseguida.

Hay que notar que un camino de s a t en la red marginal respecto a un flujo dado, corresponde a una cadena aumentante de s a t en la red original y viceversa. La capacidad incremental de una cadena será igual a la de la ruta más corta correspondiente en la red marginal. Partiendo de aquí, se establece que para determinar una cadena aumentante hay que determinar primero una ruta más corta en la red marginal y se enviará, a través de esta cadena, una cantidad igual a su capacidad incremental o igual a la diferencia entre el valor de flujo requerido y el valor del flujo actual. Así se tendrá que tratar en cada iteración, hasta alcanzar el valor requerido de *v*.

Pero podría darse caso de que en alguna iteración no exista ruta alguna de s a t en la red marginal y el valor actual de u es aun menor del *v*. En este caso se concluye que no existe ningún flujo del valor requerido debido a que no existe la ruta de s a t en la red marginal, no existen cadenas aumentantes de s a t en la red original y por lo tanto el flujo x_{ij} es flujo máximo.

En caso contrario, cuando *u* junto con la capacidad incremental de la cadena es mayor que *v* requerida, se tendrá que enviar sólo la cantidad de *v* - *u* a través de esta cadena, y así determinar el flujo deseado.

Definición. – Una ***cadena*** de s a t es ***aumentante*** si $x_{ij} < c_{ij}$ para todo $(i, j) \in C^+$ y $x_{ij} > 0$ para todo $(i,j) \in C^-$. Y se denomina como la cadena aumentante debido a que a través de ella puede enviarse flujo de s a t obteniendo así, un flujo factible de mayor valor. Donde C^+ y C^- son dos subconjuntos de arcos de C tales que: los arcos $(i_r, i_{r+1}) \in C^+$ y arcos $(i_{r+1}, i_r) \in C^-$.

Y bueno, la capacidad incremental de una cadena aumentante C es la máxima cantidad de flujo que se pueda enviar de s a t.

Resumiendo, vimos que para solucionar el problema del flujo a costo mínimo en una red, se usan una de las dos técnicas o procedimientos existentes aplicando el método simplex:

El ***algoritmo basado en eliminación de circuitos negativos***, o también se conoce como el algoritmo Primal, puesto que empieza a aplicarse a partir de un flujo factible del valor v requerido, y en cada iteración la distribución se mejora, sin modificar su valor, hasta alcanzar la optimalidad. Para ello se puede utilizarse el algoritmo de Ford y Fulkerson.

Otra técnica es basada en las ***rutas más cortas***, o también es conocido como el algoritmo Dual ya que se aplica a partir de un flujo factible de algún valor menor al de v requerido pero de costo mínimo; en cada iteración se incrementa el valor del flujo, siempre conservando la optimalidad hasta alcanzar a v.

Pero los problemas de flujo a costo mínimo en una red, igual pueden ser resueltas a través del método llamado ***método de transición de estado***. Este algoritmo es similar al algoritmo primal-dual en el sentido que empieza con factibilidad del dual, pero no necesariamente con la factibilidad del primal y se itera entre los modelos primales y duales hasta alcanzar la optimalidad. Sin embargo, la diferencia esta en que el método de transición de estado no siempre mantiene la holgura complementaria. Así que se puede ver cómo una generalización del algoritmo primal-dual para problemas de flujo en redes. Este método es el interés de este trabajo y se explicará detenidamente mas adelante

Un problema de flujo se puede transformar en un problema de flujo a costo mínimo de la siguiente manera:

Añadiendo a la red dada un arco de regreso (t, s) con una capacidad mínima (l_{ts}) igual a su capacidad máxima (c_{ts}) igual al valor de flujo dado que es valor v, o sea: $l_{ts} = c_{ts} = v$, y con costo $a_{ts} = 0$. Para todos los demás arcos (i, j) las cantidades l_{ij}, c_{ij} y a_{ij} permanecen como fueron dadas.

Ejemplo. - Si tenemos un valor de flujo $v = 19$, y la red de la figura 2.2.5 con costos:

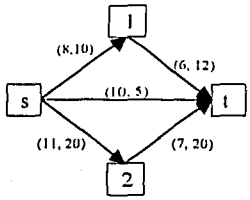


Fig 2.2.5

Transformado en un problema de flujo a costo mínimo queda de la siguiente manera:

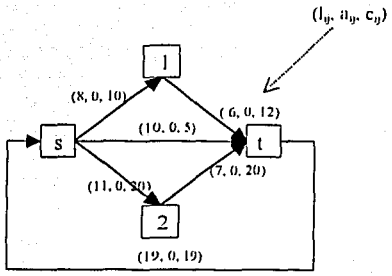


Fig 2.2.6

2.3 EL PROBLEMA DE FLUJO CON CAPACIDADES A COSTO MÍNIMO

El problema de flujo a costo mínimo con capacidades consiste en una red circular en la cual los flujos permitidos de cada arco están sujetos a cuotas inferior y superior y además se tiene un costo por unidad de flujo que pasa por cada arco. Y el objetivo que se tiene es determinar el flujo en la red con menor costo posible.

Sea G = (V, A) la red de circulación con las capacidades l_{ij} que es flujo mínimo permitido y c_{ij} el flujo máximo permitido en cada arco (i, j) ∈ A, donde se cumpla:

$$l_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\ i, j.$$

En este caso tenemos un problema de circulación con capacidades.

Si además tenemos el costo a_{ij} asignado a cada arco por unidad de flujo que pasa por éste, entonces el problema de flujo a costo mínimo es:

$$\begin{array}{lll} \text{Minimizar} & \sum_{i,j} a_{ij} x_{ij} & \\ \text{sujeto a:} & \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0 & \forall\ i \\ & l_{ij} \leq x_{ij} \leq c_{ij} & \forall\ i, j \end{array}$$

El vector de variables de decisión $x = (x_{ij})$, se llama el flujo de la red.

Muchos problemas pueden ser formulados como un problema de flujo a costo mínimo con restricciones, por ejemplo: el problema de ruta más corta, de flujo máximo, de transporte, de asignación, de transbordo, etc.

2.4 EL PROBLEMA DE FLUJO MÁXIMO

El caso especial y de gran importancia de problemas de flujos en redes son: el problema de flujo máximo y el problema de la ruta más corta. Estos problemas, como se mencionó anteriormente, se pueden resolverse mediante el método simplex o bien, por el método de transición de estado.

El problema de flujo máximo consiste en encontrar la máxima cantidad de bien que puede ser enviado entre dos puntos (el vértice fuente s y el vértice destino t). A cada arco (i, j) con el flujo se le asocian las capacidades, donde:

x_{ij} es la cantidad que fluye en el arco (i, j)
c_{ij} es la capacidad máxima que puede fluir con facilidad a través del arco (i, j)

y $0 \le x_{ij} \le c_{ij}$

Hay que notar que la cota inferior en el problema de flujo máximo es igual a cero, o sea: $l_{ij} = 0$, y se toma por hecho que las capacidades c_{ij} son enteras además en el problema de flujo máximo no intervienen los costos.

La ley de conservación se debe cumplir para todos los vértices diferentes a s y t, i.e., todo lo que entra a un vértice debe de salir

$$\sum_j x_{ji} - \sum_j x_{ij} = 0 \qquad \forall \quad i \neq s, t$$

y se tiene

$$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v & i = s \\ 0 & \forall \quad i \neq s, t \\ v & i = t \end{cases}$$

Donde cualquier conjunto de valores $x = (x_{ij})$ que satisface la ley de conservación y se cumpla la relación de $0 \le x_{ij} \le c_{ij}$, es llamado *flujo factible* o simplemente flujo, y su valor es $v = \sum_j x_{jt}$.

Se dice que un flujo es máximo, si es de valor máximo; es decir, si genera el mayor valor posible de v.

El problema de flujo máximo se puede transformar en un problema de flujo a costo mínimo de la siguiente manera:

A la red dada con vértice fuente s y vértice destino t se le agrega un arco de regreso (t, s) con capacidad mínima $l_{ts} = 0$, capacidad máxima $c_{ts} = \infty$ y un costo asociado $a_{ts} = -1$.

Para todos los demás arcos (i, j) la capacidad máxima y mínima se quedan como fueron dados y el costo $a_{ij} = 0$.[4]

En la siguiente gráfica se representa un problema de flujo máximo:
(los números asociados a los arcos en la gráfica de la figura 2.4.1, son la capacidad máxima)

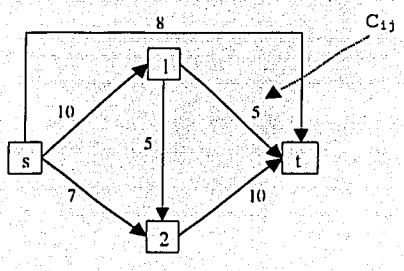


Fig 2.4.1

Donde el valor del flujo es $v = 17$.

Por lo tanto el problema anterior transformado en un problema de flujo a costo mínimo tendrá la representación gráfica como se puede ver en la figura 2.4.2:

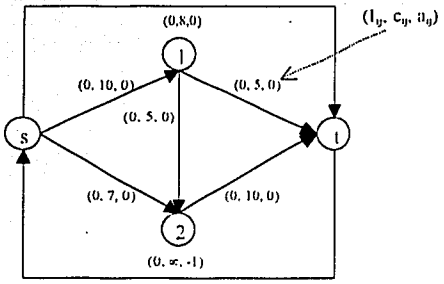


Fig 2.4.2

[4] En caso de tener un problema de flujo mínimo se hace lo mismo que con el problema de flujo máximo a excepción de que $a_{ts} = 1$

Matemáticamente el problema se plantea de siguiente manera:

Minimizar $-x_{ts}$ equivale a:

Maximizar x_{ts}
sujeto a: $x_{ts} - (x_{s1} + x_{s2} + x_{st}) = 0$

$x_{s1} - (x_{12} + x_{1t}) = 0$ $\quad 0 \le x_{s1} \le 8$ $\quad 0 \le x_{12} \le 5$
$(x_{s2} + x_{12}) - x_{2t} = 0$ $\quad 0 \le x_{s1} \le 10$ $\quad 0 \le x_{1t} \le 5$
$(x_{s1} + x_{1t} + x_{2t}) - x_{ts} = 0$ $\quad 0 \le x_{s2} \le 7$ $\quad 0 \le x_{2t} \le 10$

2.5 EL PROBLEMA DE LA RUTA MÁS CORTA

El problema de la ruta más corta se refiere a encontrar la trayectoria más corta desde un origen hasta un destino, a través de una red que los conecta dada la distancia no negativa asociada con las ramas respectivas de la red. En otras palabras, consiste en: dada una red en la cual a cada arco (i, j) se le asocia una longitud a_{ij}, encontrar la ruta más corta del vértice fuente s al vértice destino t.

Matemáticamente el problema de la ruta más corta se puede representar de la siguiente manera:

$$\text{Minimizar } \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_{ij}$$

$$\text{Sujeto a: } \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} 1 & \text{si } i=1 \\ 0 & \text{si } i \neq 1 \text{ y } i \neq m \\ -1 & \text{si } i=m \end{cases}$$

$$x_{ij} \ge 0 \qquad i=1,2,\ldots,m$$

Donde las restricciones $x_{ij} = 0$ ó 1 indica si el arco está en la ruta o no está. Como sabemos que la matriz de incidencia nodo-arco asociado con las ecuaciones de conservación de flujo es totalmente unimodular, por lo tanto, sustituyendo las $x_{ij} = 0$ ó 1 por $x_{ij} \ge 0$, igual se tendrá la solución óptima entera en la que el valor de cada variable es cero o uno.

PROBLEMA DE RUTA MÁS CORTA DEL NODO INICIAL AL FINAL

En general, en una red R = (V, A, a), al número *a* que esta asociado a cada arco (i, j) se le conose como longitud o costo del arco. Tambien, hay que notar que la longitud de una ruta o camino es la suma de las longitudes de los arcos que la forman, como lo definimos en el capitulo I. La ruta cuya longitud sea mínima se le llama ruta más corta o camino más corto.

En el caso de encontrar la ruta más corta entre dos vértices específicos *s* y *t* puede generalizarse a cualquier red ya que la función de longitud *a* puede representar, además de distancia o tiempo, costos o alguna otra cantidad.

Para que el problema de ruta más corta entre dos vértices específicos tenga solución, se deben de cumplirse los siguientes puntos:

1. Que exista un camino entre s y t,
2. Y que no existan circuitos negativos en los que haya un camino de s a cualquier otro vértice del circuito y otro de algún vértice del circuito a t. Si esto ocurre, el problema puede ser no acotado.

Si el vértice s es el origen de donde vamos a partir para llegar por el camino más corto al vértice t, que es el destino en este caso. Los diferentes caminos y sus longitudes se muestran en la red 2.5.1, siguiente:

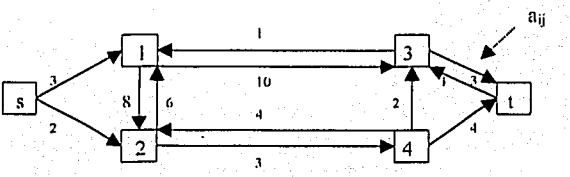


Fig 2.5.1

Este problema se puede transformar en un problema de flujo a costo mínimo de la siguiente manera:

A la red dada por el problema se le añade un arco de regreso (t, s) con su capacidad mínima igual a su capacidad máxima igual a uno, i. e., $l_{ts} = c_{ts} = 1$, y con costo igual a cero, $a_{ts} = 0$. para todos los demas arcos (i, j) se les asigna una capacidad mínima lij = 0, una capacidad máxima $c_{ij} = \infty$ y el costo a_{ij} como fue dado.

Gráficamente:

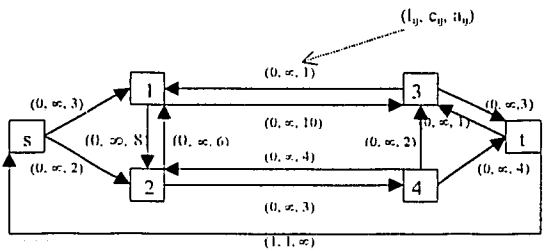


Fig 2.5.2

Y matemáticamente se representaria así:

$$\text{Minimizar } Z = 3x_{s1} + 2x_{s2} + 6x_{21} + 8x_{12} + 10x_{13} + x_{31} + 3x_{24} + 4x_{42} + 2x_{43} + 3x_{3t} + x_{t3} + 4x_{t4}$$

Sujeto a:

$$x_{ts} - (x_{s1} + x_{s2}) = 0$$
$$(x_{s1} + x_{31} + x_{21}) - (x_{12} + x_{13}) = 0$$
$$(x_{s2} + x_{12} + x_{42}) - (x_{21} + x_{24}) = 0$$
$$(x_{13} + x_{43} + x_{t3}) - (x_{31} + x_{3t}) = 0$$
$$x_{24} - (x_{42} + x_{43} + x_{4t}) = 0$$
$$(x_{3t} + x_{4t}) - (x_{t3} + x_{t5}) = 0$$

$$0 \le x_{s1} \le 3 \qquad 0 \le x_{s2} \le 2$$
$$0 \le x_{21} \le 6 \qquad 0 \le x_{12} \le 8$$
$$0 \le x_{13} \le 10 \qquad 0 \le x_{31} \le 1$$
$$0 \le x_{24} \le 3 \qquad 0 \le x_{42} \le 4$$
$$0 \le x_{43} \le 2 \qquad 0 \le x_{3t} \le 3$$
$$0 \le x_{t3} \le 1 \qquad 0 \le x_{4t} \le 4$$
$$0 \le x_{ts} \le 1$$

PROBLEMA DE RUTA MÁS CORTA DEL VÉRTICE INICIAL Y TODOS LOS DEMÁS VÉRTICES

Otro problema de ruta más corta es: dada una red encontrar la ruta más corta del vértice fuente s a todos los demás vértices de la red. Usando la misma idea de la ruta más corta entre dos vértices específicos que acabamos de ver, se puede concluir que para que exista la arborescencia de rutas más cortas de una raiz o nodo inicial s en una red R = (V, A, a), se debe de cumplirse:

1. Que existan caminos de s a cualquier nodo, i.e., que s sea la raiz de la red.
2. Que no existan circuitos negativos en la red R cuya presencia indicaría que el problema no esta acotado.

Este problema se puede expresar como un problema de flujo a costo mínimo de la siguiente manera:

Añadiéndole a la red arcos de regreso (j, s) de todos los vértices $j \neq s$ con su capacidad mínima igual a su capacidad máxima igual a uno y con costo cero, i.e., $l_{js} = c_{js} = 1$ y $a_{js} = 0$.

Suponiendo que en la siguiente red (fig 2.5.3), se quiera encontrar ruta más corta de un vértice s a todos los demás vértices de la red, y la red transformada a un problema de circulación con costo mínimo.

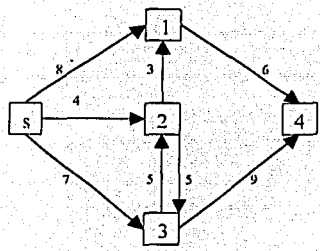


Fig 2.5.3

El problema resultante sería:

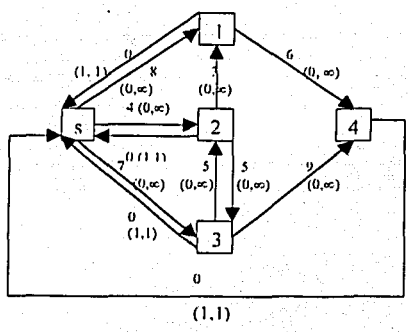


Fig 2.5.4

2.6 EL MÉTODO DE TRANSICIÓN DE ESTADO

Como ya lo habíamos comentado, para resolver los problemas de programación lineal que son problemas de flujo con costo mínimo se puede usar el método de transición de estado. Este método resuelve problemas de flujo sin modificar la estructura de red original. Es un algoritmo general del método prima-dual para los problemas de flujo en redes.

El algoritmo out-of-kilter puede iniciar con cualquier clase de flujo (positivo o negativo) en la red. Si este es factible, entonces por medio del algoritmo se va a convertir en óptimo, y si no es así, entonces, primero lo convertirá en factible y después en óptimo. Por su estructura especial, el algoritmo requiere que las capacidades sean números enteros en todos los arcos y que la red a tratar sea circular.

Recordemos que el problema de flujo a costo mínimo consiste en:

$$\text{Minimizar} \sum_{i,j} a_{ij} x_{ij}$$

$$\text{sujeto a:} \quad \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0 \qquad \forall\, i$$

$$0 \le l_{ij} \le x_{ij} \le c_{ij} \qquad \forall\, i, j$$

Donde las sumatorias y desigualdades se toman únicamente sobre los arcos existentes. Como definimos en el capítulo uno, un flujo circulatorio que satisface las restricciones

$$l_{ij} \le x_{ij} \le c_{ij}$$

es un flujo o solución factible, son enteros y se respeta la relación:

$$0 \le l_{ij} \le c_{ij}.$$

Hay que notar que como todos los valores del lado derecho de las restricciones son ceros, entonces el flujo en la red no va a tener nodo inicial ni nodo final, sino que más bien todo el flujo circulatorio en la red será a lo largo de circuitos (ciclos dirigidos).

Este problema se puede expresar como un problema de programación lineal de la siguiente manera:

$$\text{Minimizar } z = \bar{a}^{T} \cdot x$$

$$\text{sujeto a:} \quad \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0 \qquad \forall\, i$$

$$x_{ij} \ge l_{ij} \qquad \forall\, i, j$$

$$-x_{ij} \ge -c_{ij} \qquad \forall\, i, j$$

Si en el modelo general del problema de flujo a costo mínimo que es el siguiente:

$$\text{Minimizar} \sum_{i,j} a_{ij} x_{ij}$$

$$\text{sujeto a:} \quad \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0 \qquad \forall\, i$$

$$0 \le l_{ij} \le x_{ij} \le c_{ij} \qquad \forall\, i, j$$

Le asociamos una variable dual u_i con cada ecuación de la conservación y una variable dual γ_{ij} con las restricciones $x_{ij} \le c_{ij}$ junto con la otra variable dual λ_{ij} con las restricciones de $x_{ij} \ge l_{ij}$, entonces el dual del método de transición de estado para el problema de flujo a costo mínimo en una red se representa de siguiente forma general:

$$\text{Maximizar} \sum_{i=1}\sum_{j=1} l_{ij}\,\lambda_{ij} - \sum_{i=1}\sum_{j=1} c_{ij}\,\gamma_{ij}$$

$$\text{sujeto a:} \quad u_i - u_j + \lambda_{ij} - \gamma_{ij} = a_{ij} \qquad \forall\, i, j$$

$$\lambda_{ij} \ge 0$$

$$\gamma_{ij} \ge 0$$

u_i no restringida,

Donde u_i es variable dual asociado con cada ecuación de conservación en nodo i.
Y las λ_{ij} y γ_{ij} están identificadas con las restricciones primales $x_{ij} \ge l_{ij}$ y $-x_{ij} \ge -c_{ij}$ (por el modelo dual).

Las sumas y las restricciones del modelo se toman sobre los arcos existentes. Al seleccionar cualquier conjunto de las variables duales u_i (que son enteras) la restricción dual para un arco espesifico será:

$$\lambda_{ij} - \gamma_{ij} = a_{ij} - u_i + u_j \qquad \text{donde} \qquad \lambda_{ij} \ge 0 \text{ y } \gamma_{ij} \ge 0$$

y esto se satisface si:

$$\lambda_{ij} = \text{Max}\,(0,\, a_{ij} - u_i + u_j)$$
$$\gamma_{ij} = \text{Max}\,(0,\, -(a_{ij} - u_i + u_j))$$

Así, dado cualquier conjunto de las u_i, el problema dual siempre tendrá una solución factible.

En el método de transición de estado tenemos que considerar las condiciones de holgura de la siguiente manera:

$$(x_{ij} - l_{ij})\,\lambda_{ij} = 0 \qquad i,j = 1, 2, \ldots, m$$
$$(c_{ij} - x_{ij})\,\gamma_{ij} = 0 \qquad i,j = 1, 2, \ldots, m$$

como $z_{ij} - c_{ij} \equiv u_i - u_j - c_{ij}$, entonces se tiene que:

$$\lambda_{ij} = \text{máx}\,\{0, -(u_i - u_j)\}$$
$$\gamma_{ij} = \text{máx}\,\{0, u_i - u_j\}$$

Es equivalente a que el problema tuviera la función objetivo con restricciones, o esa, con cotas superior y inferior. Pero en este caso no es necesario tener una solución básica. Las condiciones que acabamos de plantear son ciertas sólo si:

$$u_i - u_j < 0 \Rightarrow \lambda_{ij} > 0 \Rightarrow x_{ij} = l_{ij} \qquad \forall\ i, j = 1, 2, ..., m$$

y

$$u_i - u_j > 0 \Rightarrow \gamma_{ij} > 0 \Rightarrow x_{ij} = u_{ij} \qquad \forall\ i, j = 1, 2, ..., m$$
$$u_i - u_j = 0 \Rightarrow l_{ij} \le x_{ij} \le c_{ij} \qquad \forall\ i, j = 1, 2, ..., m$$

Si estas condiciones se satisfacen, entonces el problema de red con ese flujo será óptimo, y podemos concluir que para obtener la solución óptima en un flujo circular se tienen que obtenerse los valores de las u_i entre las x_{ij} hasta que las condiciones de holgura se satisfagan.

Si se tiene el siguiente problema de flujo a costo mínimo en su forma primal como se muestra en la figura 2.6.1:

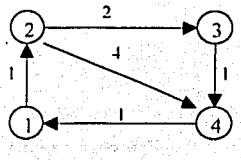


Fig 2.6.1

Minimizar $z = x_{12} + 2x_{23} + 4x_{24} + x_{34} + x_{41}$

sujeto a:

$$\left.\begin{aligned} -x_{12} + \qquad\qquad + x_{41} &= 0 \\ x_{12} - x_{23} - x_{24} \qquad\quad &= 0 \\ x_{23} - x_{34} \qquad &= 0 \\ x_{24} + x_{34} - x_{41} &= 0 \end{aligned}\right\}$$ ley de conservación

$$\left.\begin{aligned} x_{12} &\ge 1 \\ x_{23} &\ge 0 \\ x_{24} &\ge 1 \\ x_{34} &\ge 2 \\ x_{41} &\ge 0 \end{aligned}\right\}$$ restricciones de cota inferior

$$\left.\begin{aligned} -x_{12} &\ge -3 \\ -x_{23} &\ge -2 \\ -x_{24} &\ge -4 \\ -x_{34} &\ge -3 \\ -x_{41} &\ge -3 \end{aligned}\right\}$$ restricciones de cota superior

Y si expresamos este problema de flujo en forma matricial, tomando las restricciones de cota inferior primero, seguidas por las restricciones de cota superior y finalmente las de la ley de conservación, tenemos:

$$A = \begin{array}{c} \begin{matrix} x_{12} & x_{23} & x_{24} & x_{34} & x_{41} \end{matrix} \\ \left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{array}\right) \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \} = A^1 \\ \\ \\ \\ \\ \\ \} = A^2 \end{array}$$

Restricciones de Cota inferior

Restricciones de Cota superior

Restricciones de la Ley de Conservación

Hay que hacer notar que la forma matricial de las restricciones de cota inferior es la matriz identidad, la de las restricciones de cota superior es el negativo de la matriz identidad y de la ley de conservación es la matriz de coeficientes de la gráfica que representa al problema.

$$b = \begin{array}{l} \left(\begin{array}{c} 1 \\ 0 \\ 1 \\ 2 \\ 0 \\ -3 \\ -2 \\ -4 \\ -3 \\ -3 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right) \end{array} \begin{array}{l} \} b^1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \} b^2 \end{array} \qquad x = \begin{array}{l} \left(\begin{array}{c} x_{12} \\ x_{23} \\ x_{24} \\ x_{34} \\ x_{41} \end{array}\right) \end{array} \} x^1 \qquad c = \left(\begin{array}{c} 1 \\ 2 \\ 4 \\ 1 \\ 1 \end{array}\right)$$

Donde m es el número de las variables de las restricciones de cota superior, inferior y de la ley de conservación, compone el vector b y es igual a 14; n es el número de las variables del problema (los arcos) esta representado por el vector x y es igual a 5; p es el número de las variables de las restricciones de cota superior y inferior está en el vector b^1 y es igual a 10; q representa el vector de los coeficientes de la función objetivo del problema, se representa a través del vector c y es igual a 5.

Para pasar al problema dual, como n = q, $A^3 = A$ y $\bar{c}^1 = \bar{c}$, y si tomamos al vector π que es variable dual asociado con cada ecuación de conservación en nodo i de la siguiente forma:

$$\pi = \begin{pmatrix} \lambda_{12} \\ \lambda_{23} \\ \lambda_{24} \\ \lambda_{34} \\ \lambda_{41} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \\ \gamma_{41} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \pi^1 \\ \\ \\ \pi^2 \end{matrix}$$

Y si ahora calculamos $(A^3)^T \pi =$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \lambda_{12} \\ \lambda_{23} \\ \lambda_{24} \\ \lambda_{34} \\ \lambda_{41} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \\ \gamma_{41} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} \lambda_{12} - \gamma_{12} - u_1 + u_2 \\ \lambda_{23} - \gamma_{23} - u_2 + u_3 \\ \lambda_{24} - \gamma_{24} - u_2 + u_4 \\ \lambda_{34} - \gamma_{34} - u_3 + u_4 \\ \lambda_{41} - \gamma_{41} - u_4 + u_1 \end{pmatrix}$$

Con las matrices y vectores anteriores podemos expresar el problema dual de la siguiente manera:

Maximizar $w = \lambda_{12} + \lambda_{24} + 2\lambda_{34} - 3\gamma_{12} - 2\gamma_{23} - 4\gamma_{24} - 3\gamma_{34} - 3\gamma_{41}$

sujeto a:

$$\lambda_{12} - \gamma_{12} - u_1 + u_2 \le 1$$
$$\lambda_{23} - \gamma_{23} - u_2 + u_3 \le 2$$
$$\lambda_{24} - \gamma_{24} - u_2 + u_4 \le 4$$
$$\lambda_{34} - \gamma_{34} - u_3 + u_4 \le 1$$
$$\lambda_{41} - \gamma_{41} - u_4 + u_1 \le 1$$

$\lambda_{ij} \geq 0$
$\gamma_{ij} \geq 0$
u_i no restringida

Aplicando el teorema de ortogonalidad de soluciones óptimas al problema de circulación con costo mínimo tenemos:

$$\begin{aligned}(u_j - u_i + \lambda_{ij} - \gamma_{ij} - a_{ij})x_{ij} &= \lambda_{ij}(x_{ij} - l_{ij}) \\ &= \gamma_{ij}(c_{ij} - x_{ij}) \\ &= 0\end{aligned}$$

esto es cierto sí, y solo si,
para

$$\begin{aligned}x_{ij} > 0 &\rightarrow u_j - u_i + \lambda_{ij} - \gamma_{ij} = a_{ij} \\ \lambda_{ij} > 0 &\rightarrow x_{ij} = l_{ij} \\ \gamma_{ij} > 0 &\rightarrow x_{ij} = c_{ij}\end{aligned}$$

Estas condiciones son equivalentes a las condiciones de buen estado, las cuales se definen como:

$$\begin{aligned}x_{ij} = l_{ij} &\Rightarrow u_j - u_i \leq a_{ij} \\ l_{ij} < x_{ij} < c_{ij} &\Rightarrow u_j - u_i = a_{ij} \\ x_{ij} = c_{ij} &\Rightarrow u_j - u_i \geq a_{ij}\end{aligned}$$

A continuación vamos a ver la equivalencia que existe entre estos dos conjuntos de condiciones primal y dual:

Sea $x = (x_{ij})$ una solución primal; y para cada arco (i, j) tenemos:

- Si $0 < l_{ij} = x_{ij} < c_{ij}$

tenemos que $x_{ij} > 0 \Rightarrow u_j - u_i + \lambda_{ij} - \gamma_{ij} = a_{ij}$,
pero $x_{ij} < c_{ij} \Rightarrow \gamma_{ij} = 0 \Rightarrow u_j - u_i + \lambda_{ij} = a_{ij}$

$\Rightarrow u_j - u_i \leq a_{ij}$

- Si $l_{ij} < x_{ij} < c_{ij}$,

tenemos que $x_{ij} > 0 \Rightarrow u_j - u_i + \lambda_{ij} - \gamma_{ij} = a_{ij}$,

pero $l_{ij} < x_{ij} \Rightarrow \lambda_{ij} = 0$
$x_{ij} < c_{ij} \Rightarrow \gamma_{ij} = 0$
$\Rightarrow u_j - u_i = a_{ij}$

- Si $0 \leq l_{ij} < x_{ij} = c_{ij}$,

tenemos que $x_{ij} > 0 \Rightarrow u_j - u_i + \lambda_{ij} - \gamma_{ij} = a_{ij}$,
pero $l_{ij} < x_{ij} \Rightarrow \lambda_{ij} = 0$

$\Rightarrow u_j - u_i \geq a_{ij}$

Sean λ_{ij} y γ_{ij} una solución dual; entonces para cada arco (i, j) tenemos:

* $\lambda_{ij} > 0 \Rightarrow x_{ij} = l_{ij}$ (1)
* $\lambda_{ij} = \gamma_{ij} = 0$ (2)
* $\gamma_{ij} > 0 \Rightarrow x_{ij} = c_{ij}$ (3)

Las condiciones de buen estado se cumplen si, y solo si, la solución primal y la solución dual son óptimas.

Las condiciones que acabamos de describir las podemos resumir en la tabla 2:

	$x_{ij} > c_{ij}$	$x_{ij} = c_{ij}$	$l_{ij} < x_{ij} < c_{ij}$	$x_{ij} = l_{ij}$	$x_{ij} < l_{ij}$,
$u_j - u_i \le a_{ij}$	$\lvert x_{ij} - l_{ij} \rvert$	$\lvert x_{ij} - l_{ij} \rvert$	$\lvert x_{ij} - l_{ij} \rvert$	0	$\lvert x_{ij} - l_{ij} \rvert$
$u_j - u_i = a_{ij}$	$x_{ij} - c_{ij}$	0	0	0	$l_{ij} - x_{ij}$
$u_j - u_i \ge a_{ij}$	$\lvert x_{ij} - c_{ij} \rvert$	0	$\lvert x_{ij} - c_{ij} \rvert$	$\lvert x_{ij} - c_{ij} \rvert$	$\lvert x_{ij} - c_{ij} \rvert$

Tabla 2

Las condiciones de buen estado se representan para cada arco en *diagrama de buen estado* (fig 2.6.2). Los puntos $(x_{ij}, u_j - u_i)$ que están en la línea quebrada son puntos que se encuentran en buen estado, y los puntos que no están sobre la línea quebrada son puntos que están en mal estado, como se puede apreciar en la siguiente figura:

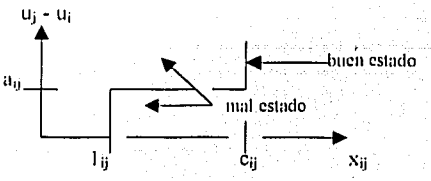


Fig 2.6.2

A cada punto $(x_{ij}, u_j - u_i)$ le asignamos un *número de buen estado*, $K(x_{ij})$, igual al valor absoluto del cambio en x_{ij} necesario para poner el arco (i, j) en buen estado. Así,

$$K(x_{ij}) = \begin{cases} \lvert x_{ij} - l_{ij} \rvert, & \text{si } u_j - u_i < a_{ij} \\ l_{ij} - x_{ij}, & \text{si } x_{ij} < l_{ij} \text{ y } u_j - u_i = a_{ij} \\ x_{ij} - c_{ij}, & \text{si } x_{ij} > c_{ij} \text{ y } u_j - u_i = a_{ij} \\ 0, & \text{si } l_{ij} < x_{ij} < c_{ij} \text{ y } u_j - u_i = a_{ij} \\ \lvert x_{ij} - c_{ij} \rvert, & \text{si } u_j - u_i > a_{ij} \end{cases}$$

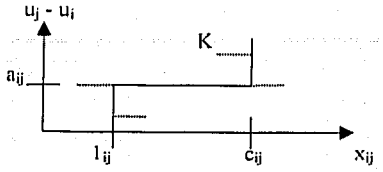


Fig 2.6.3

En la gráfica de la figura 2.6.3, las líneas punteadas indican el posible cambio K en los arcos para alcanzar buen estado.

El objetivo del método de transición de estado es obtener una circulación $x = (x_{ij})$ y un conjunto de números de nodos $u = (u_{ij})$ para los cuales las condiciones de buen estado se satisfacen.

Como las condiciones de buen estado se satisfacen si, y solo si, todos los números de buen estado son cero, la suma de los números de buen estado es $\sum_{i,j} K(x_{ij})$, puede ser usada como una medida de mejoría alrededor de un par de soluciones.

Los cálculos del método de transición de estado se empiezan calculando una circulación, factible o no, pero teniendo cuidado de que la ley de conservación se cumplan en todos los vértices y con cualquier conjunto de números de nodos.

Supongamos que se tiene el problema representado en la siguiente red (Fig 2.6.4), vamos a obtener la solución usando el método de transición de estado:

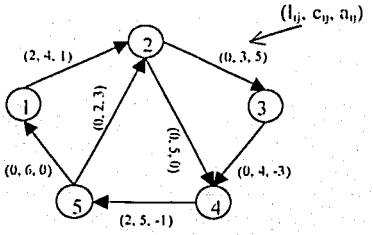


Fig 2.6.4

Iniciando con el flujo circulatorio $x_{ij} = 0$ y una solución factible para el problema dual $u_i = 0$, se obtiene la siguiente gráfica:

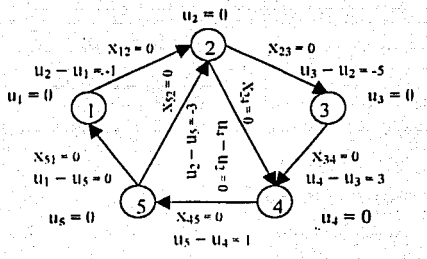


Fig 2.6.5

Analizando cada uno de los arcos de la red, se determinan sus estados:

$u_2 - u_1 = -1$	<	0	→	$x_{12} = 0$	≠	$l_{12} = 2$	∴	m.e.[5]
$u_3 - u_2 = -5$	<	0	→	$x_{23} = 0$	=	$l_{23} = 0$	∴	b.e.
$u_4 - u_3 = 3$	>	0	→	$x_{34} = 0$	≠	$c_{34} = 4$	∴	m.e.
$u_5 - u_4 = 1$	>	0	→	$x_{45} = 0$	≠	$c_{45} = 5$	∴	m.e.
$u_1 - u_5 = 0$	=	0	→	$x_{51} = 0$	≥	$l_{51} = 0$		
		0		$x_{51} = 0$	≤	$c_{51} = 6$	∴	m.e.
$u_2 - u_5 = -3$	<	0	→	$x_{52} = 0$	=	$l_{52} = 0$	∴	b.e.
$u_4 - u_2 = 0$	=	0	→	$x_{24} = 0$	≥	$l_{24} = 0$		
		0	→	$x_{24} = 0$	≤	$c_{24} = 5$	∴	b.e.

Podemos ver que el flujo del arco (1, 2) tiene la diferencia de 2 y puede ser aumentado, los arcos (2, 3) y (5, 2) no tendrán cambios de flujo, por lo tanto los podemos omitir en la construcción de G', fig 2.6.6:

G'

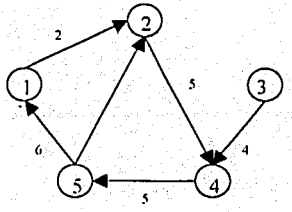


Fig 2.6.6

[5] donde m.e. y b.e. representan mal y buen estado del arco respectivamente.

(t, s) = (1, 2)
Circuito = {(1, 2), (2, 4), (4, 5), (5, 1)}
$\varphi = \min\{2, 5, 5, 6\} = 2$

Cambiando el flujo actual se tiene la red G de flujo fig 2.6.7:

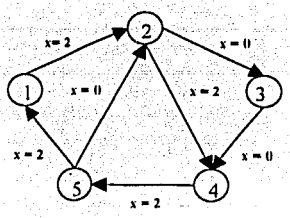


Fig 2.6.7

Se verifica de nuevo el estado de los arcos de la red con el propósito de asegurarse que no haya los cambios en los arcos con buen estado. Y para asegurarse de que el arco (t, s) ya tiene buen estado.

$u_2 - u_1 = -1$	$<$	0	$\rightarrow$	$x_{12} = 2$	$=$	$l_{12} = 2$	$\therefore$	b.e.	Se omite en G'
$u_3 - u_2 = -5$	$<$	0	$\rightarrow$	$x_{23} = 0$	$=$	$l_{23} = 0$	$\therefore$	b.e.	Se omite en G'
$u_4 - u_3 = 3$	$>$	0	$\rightarrow$	$x_{34} = 0$	$\neq$	$c_{34} = 4$	$\therefore$	m.e.	
$u_5 - u_4 = 1$	$>$	0	$\rightarrow$	$x_{45} = 0$	$\neq$	$c_{45} = 5$	$\therefore$	m.e.	
$u_1 - u_5 = 0$	$=$	0	$\rightarrow$	$x_{51} = 0$	$\geq$	$l_{51} = 0$			
		0		$x_{51} = 0$	$\leq$	$c_{51} = 6$	$\therefore$	m.e.	
$u_2 - u_5 = -3$	$<$	0	$\rightarrow$	$x_{52} = 0$	$=$	$l_{52} = 0$	$\therefore$	b.e.	Se omite en G'
$u_4 - u_2 = 0$	$=$	0	$\rightarrow$	$x_{24} = 0$	$\geq$	$l_{24} = 0$			
		0	$\rightarrow$	$x_{24} = 0$	$\leq$	$c_{24} = 5$	$\therefore$	b.e.	

Como podemos ver, los arcos (5, 2) y (1, 2) son los que no van a tener ningún cambio de flujo y por lo tanto los omitiremos en la siguiente gráfica G'.

La red de cambio de flujo G' respectiva se muestra en la figura 2.6.8:

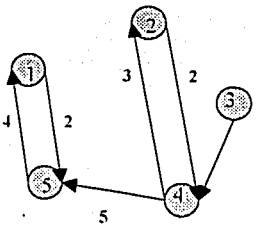


Fig 2.6.8

(r, s) = (3, 4)
Circuito = {(3, 4), (4, 5), (5, 1)}
$\varphi = \min\{0, 2, 2\} = 2$

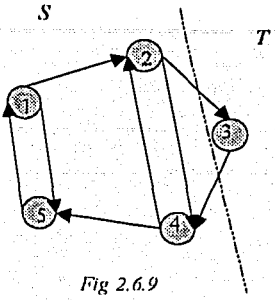


Fig 2.6.9

$S = \{5, 1, 2, 4\}$, $T = \{3\}$
$F = \{(2, 3)\}$, $M = \{(3, 4)\}$
$\varepsilon_1 = \min\{|-5|\} = 5$
$\varepsilon_2 = \min\{|3|\} = 3$
$\varepsilon = \min\{5, 3\} = 3$

Actualizando las u_i 's y $(u_j - u_i)$:

$u_1' = u_1 + 3 = 3$
$u_2' = u_2 + 3 = 3$
$u_3' = u_3 = 0$
$u_4' = u_4 + 3 = 3$
$u_5' = u_5 + 3 = 3$

$(u_3 - u_2)' = (u_3 - u_2) + \varepsilon \quad \rightarrow \quad -5 + 3 = -2$
$(u_4 - u_3)' = (u_4 - u_3) - \varepsilon \quad \rightarrow \quad 3 - 3 = 0$

Gráficamente:

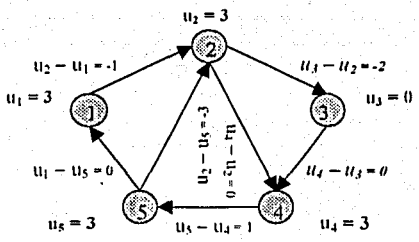


Fig 2.6.10

El arco sobre el cual estábamos trabajando (3, 4) ya está en buen estado, por lo tanto después de verificarlo por medio de las condiciones de buen estado, trabajaremos con el último arco que se encuentra en mal estado (4, 5).

$u_2 - u_1 = -1$	<	0	→	$x_{12} = 2$	=	$l_{12} = 2$	∴	b.e.
$u_3 - u_2 = -2$	<	0	→	$x_{23} = 0$	=	$l_{23} = 0$	∴	b.e.
$u_4 - u_3 = 0$	=	0	→	$x_{34} = 0$	≥	$l_{34} = 0$		
				$x_{34} = 0$	≤	$c_{34} = 4$	∴	b.c.
$u_5 - u_4 = 1$	>	0	→	$x_{45} = 2$	≠	$c_{45} = 5$	∴	m.e.
$u_1 - u_5 = 0$	=	0	→	$x_{51} = 2$	≥	$l_{51} = 0$		
		0		$x_{51} = 2$	≤	$c_{51} = 6$	∴	b.e.
$u_2 - u_5 = -3$	<	0	→	$x_{52} = 0$	=	$l_{52} = 0$	∴	b.e.
$u_4 - u_2 = 0$	=	0	→	$x_{24} = 2$	≥	$l_{24} = 0$		
		0	→	$x_{24} = 2$	≤	$c_{24} = 5$	∴	b.e.

(t, s) = (4, 5)

S = {1, 5}, T = {2, 4, 3}
F = {(1, 2). (5, 2)}, M = {(4, 5)}
$\varepsilon_1 = \min \{|-1, -3|\} = 1$
$\varepsilon_2 = \min \{|1|\} = 1$
$\varepsilon = \min \{1, 1\} = 1$

G'

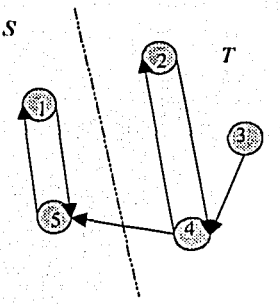


Fig 2.6.11

$u_1' = u_1 + 1 = 4$
$u_2' = u_2 = 3$
$u_3' = u_3 = 0$
$u_4' = u_4 = 3$
$u_5' = u_5 + 1 = 4$
$(u_2 - u_1)' = (u_2 - u_1) + \varepsilon \rightarrow -1 + 1 = 0$
$(u_2 - u_5)' = (u_2 - u_5) + \varepsilon \rightarrow -3 + 1 = -2$
$(u_5 - u_4)' = (u_5 - u_4) - \varepsilon \rightarrow 1 - 1 = 0$

Graficamente:

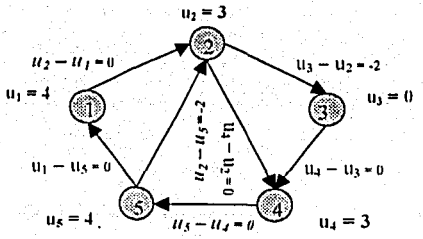


Fig 2.6.12

Aplicando el método de transición de estado resolvimos el problema de la red dada, ya que no existen más arcos en mal estado obtuvimos la solución óptima siguiente:

$x_{12} = 2$ $\quad x_{51} = 2$
$x_{23} = 0$ $\quad x_{52} = 0$
$X_{34} = 0$ $\quad x_{24} = 2$
$x_{45} = 2$ $\quad Z = -2$

ALGORITMO DEL MÉTODO DE TRANSICIÓN DE ESTADO *ref. (9)* [6]

Podemos resumir y plantear el algoritmo de transición de estado en tres etapas que son: inicial, primal y dual, de la siguiente manera:

Etapa Inicial	1) Se comienza con flujo (entero) circulatorio: $x_{ij} = 0$ y un conjunto de variables duales (enteras): $u_i = 0$ 2) Se calcula: $u_i - u_j - a_{ij}$
Etapa Primal	1) Se determina el estado de cada arco de la red, si todos los arcos están en buen estado, implica que ya se tiene la solución óptima. En caso contrario, 2) Se selecciona un arco (t, s) en mal estado y 3) Se construye la subgráfica G' 4) A partir de la gráfica G' se busca algún circuito que contenga dicho arco (t, s) y se determina φ igual al mínimo en cambio de flujo. 5) Se hace el cambio en el flujo de la red según las siguientes reglas: $x'_{ij} = \begin{cases} x_{ij} + \varepsilon & \text{si } (i, j) \in G' \\ x_{ij} - \varepsilon & \text{si } (i, j) \in G' \\ x_{ij} & \text{si no está en } G' \end{cases}$ 6) Se repite la fase primal 7) En caso de que no exista en G' ningún circuito con el arco (t, s), entonces se pasa a la fase dual.

[6] La presentación del algoritmo está basado en Clasen (9) y la esencia, fue presentada por Jewell (10)

Etapa Dual	1) Se determinan el conjunto S (es un conjunto de nodos en G' que se puedan alcanzar desde nodo s), y el conjunto T = N – S, donde N es el número total de los nodos. 2) Se definen F y M: $F = \{(i,j): i \in S, j \in T, u_i - u_j < 0, x_{ij} \leq c_{ij}\}$ $M = \{(i,j): i \in T, j \in S, u_i - u_j > 0, x_{ij} \geq l_{ij}\}$ 3) Se calculan ε: $\varepsilon_1 = \min \{| u_i - u_j |\}$ $(i, j) \in F$ $\varepsilon_2 = \min \{| u_i - u_j |\}$ $(i, j) \in M$ $\varepsilon = \min \{| \varepsilon_1, \varepsilon_2 |\}$ En caso de que $\varepsilon = \infty$, no existe la solución factible del problema. 4) Se hace el cambio de las variables duales u_i y los $u_i - u_j$ según: $(u_i - u_j)' = \begin{cases} (u_i - u_j) & \text{si } (i,j) \in (S, S) \cup (T, T) \\ (u_i - u_j) + \varepsilon & \text{si } (i,j) \in (S, T) \\ (u_i - u_j) - \varepsilon & \text{si } (i,j) \in (T, S) \end{cases}$ 5) Se pasa a la fase primal.

2.7 MÉTODO DE TRANSICIÓN DE ESTADO POR MINTY

El método de transición de estado usando el teorema de coloración es el propuesto por Minty. El algoritmo tiene la misma base, y las condiciones de buen estado que son:

$$\begin{aligned} x_{ij} = l_{ij} &\Rightarrow u_j - u_i \leq a_{ij} \\ l_{ij} < x_{ij} < c_{ij} &\Rightarrow u_j - u_i = a_{ij} \\ x_{ij} = c_{ij} &\Rightarrow u_j - u_i \geq a_{ij} \end{aligned}$$

Pero a partir del teorema de Coloración que se explica en el primer capítulo.

En cada iteración se hace un cambio, ya sea en la circulación o bien en el número de nodos de la red. El tipo de cambio esta determinado por la aplicación del teorema de coloración de Minty.

A continuación veremos como se propone pintar los arcos y cambiar las direcciones de algunos de ellos para poder aplicar el teorema de coloración de Minty:

Pintar el arco de *verde* si éste esta en buen estado y es posible incrementar o disminuir el flujo del arco sin que éste pase a estar en mal estado.

$$l_{ij} < x_{ij} \leq c_{ij} \quad y \quad u_j - u_i = a_{ij}$$

Pintar el arco de *amarillo* si es posible incrementar el flujo del arco pero no disminuirlo sin que aumente el número de buen estado del arco.

$$\begin{aligned} x_{ij} < c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} < l_{ij} \quad & y \quad u_j - u_i = a_{ij} \\ x_{ij} < l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

Pintar el arco de *amarillo** e invertir su dirección, si es posible disminuir el flujo del arco pero no aumentarlo sin incrementar el número de buen estado del arco.

$$\begin{aligned} x_{ij} > c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} \geq c_{ij} \quad & y \quad u_j - u_i = a_{ij} \\ x_{ij} > l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

Pintar el arco de *rojo* si el flujo de éste no puede ser aumentado ni disminuido sin que aumente el número de buen estado del arco.

$$\begin{aligned} x_{ij} = c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} = l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

2.7 MÉTODO DE TRANSICIÓN DE ESTADO POR MINTY

El método de transición de estado usando el teorema de coloración es el propuesto por Minty. El algoritmo tiene la misma base, y las condiciones de buen estado que son:

$$\begin{aligned} x_{ij} = l_{ij} &\Rightarrow u_j - u_i \leq a_{ij} \\ l_{ij} < x_{ij} < c_{ij} &\Rightarrow u_j - u_i = a_{ij} \\ x_{ij} = c_{ij} &\Rightarrow u_j - u_i \geq a_{ij} \end{aligned}$$

Pero a partir del teorema de Coloración que se explica en el primer capitulo.

En cada iteración se hace un cambio, ya sea en la circulación o bien en el número de nodos de la red. El tipo de cambio esta determinado por la aplicación del teorema de coloración de Minty.

A continuación veremos como se propone pintar los arcos y cambiar las direcciones de algunos de ellos para poder aplicar el teorema de coloración de Minty:

Pintar el arco de *verde* si éste esta en buen estado y es posible incrementar o disminuir el flujo del arco sin que éste pase a estar en mal estado.

$$l_{ij} < x_{ij} z\, c_{ij} \quad y \quad u_j - u_i = a_{ij}$$

Pintar el arco de *amarillo* si es posible incrementar el flujo del arco pero no disminuirlo sin que aumente el número de buen estado del arco.

$$\begin{aligned} x_{ij} < c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} < l_{ij} \quad & y \quad u_j - u_i = a_{ij} \\ x_{ij} < l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

Pintar el arco de *amarillo** e invertir su dirección, si es posible disminuir el flujo del arco pero no aumentarlo sin incrementar el número de buen estado del arco.

$$\begin{aligned} x_{ij} > c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} \geq c_{ij} \quad & y \quad u_j - u_i = a_{ij} \\ x_{ij} > l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

Pintar el arco de *rojo* si el flujo de éste no puede ser aumentado ni disminuido sin que aumente el número de buen estado del arco.

$$\begin{aligned} x_{ij} = c_{ij} \quad & y \quad u_j - u_i > a_{ij} \\ x_{ij} = l_{ij} \quad & y \quad u_j - u_i < a_{ij} \end{aligned}$$

En el diagrama de la figura 2.7.1, se ve gráficamente como se colorean los arcos de una red

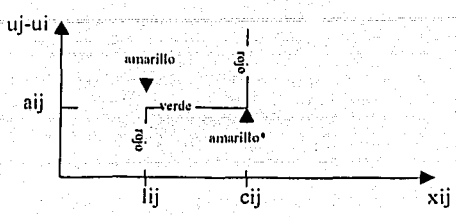


Fig 2.7.1

Vamos a analizar el arco amarillo (t, s) en mal estado. Si encontramos un ciclo amarillo-verde modificamos la circulación alrededor del ciclo. Si encontramos un enlace amarillo-rojo usaremos este enlace como una base para revisar el número de nodos.

Supongamos que hay un ciclo amarillo-verde, C, en el cual todos los arcos amarillos están orientados en la misma dirección que (t, s). Reorientando todos los arcos cuyas direcciones fueron invertidas cuando fueron pintadas de amarillo*. Un incremento en una pequeña cantidad $\delta > 0$ en el flujo a través del arco (t, s) disminuirá su número de buen estado en una pequeña cantidad, suponiendo que el número de buen estado es finito.

Un incremento en δ en el flujo a través de los arcos de C orientados en la misma dirección que (t, s) y una disminución en δ en los otros arcos, no incrementara el número de buen estado de ningún arco, y tal vez disminuirá los números de buen estado de algunos de los arcos, i.e., C – (t, s) describe una trayectoria de flujo aumentado.

Veamos un ejemplo donde se tiene el ciclo *amarillo-verde* como en la figura 2.7.2, donde *v* y *a* representan un arco de color verde y amarillo respectivamente:

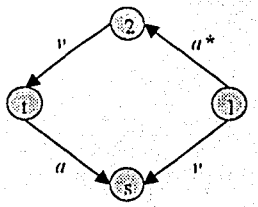


Fig 2.7.2

Reorientando los arcos amarillo*, y aumentando y disminuyendo en δ el flujo de los arcos del ciclo como se describió anteriormente tenemos la siguiente figura 2.7.3, en la cual la suma de los números de buen estado disminuye:

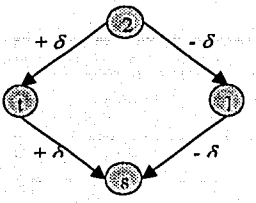


Fig 2.7.3

Ahora se van a construir y analizar las diagramas de buen estado de cada uno de los arcos del ciclo.

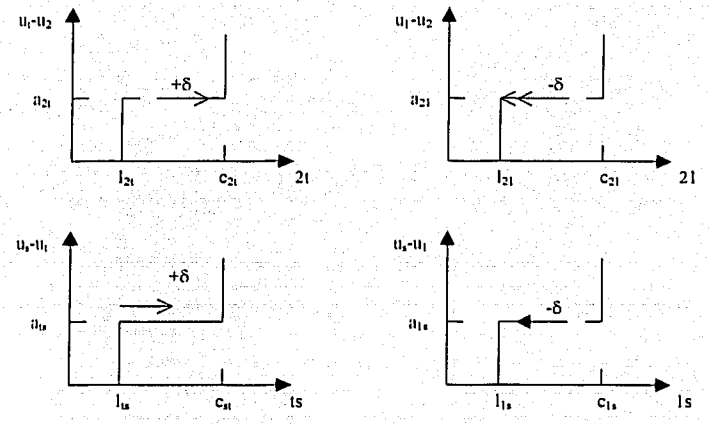


Fig 2.7.4

Después de analizar los arcos uno por uno se construye un diagrama de buen estado (fig 2.7.5) con todos los cambios en δ posibles para los arcos de un ciclo amarillo-verde:

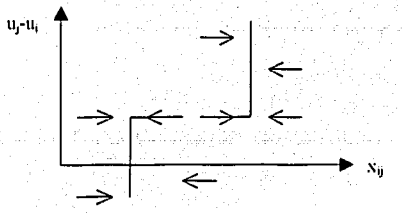


Fig 2.7.5

Como se puede ver en la figura anterior, no hay incremento en el número de buen estado del arco siempre que en δ sea lo suficientemente pequeña. A continuación veremos de que manera se tiene que escoger la δ.

Para un ciclo C amarillo-verde dado, sean Y y G los subconjuntos de C de arcos amarillos y verdes respectivamente. Les vamos a asignar los superíndices + y – para indicar los subconjuntos de Y y G para los cuales el flujo del arco va a ser respectivamente incrementado o disminuido en δ.

Ningún arco en buen estado pasará a mal estado si δ no es más grande que δ_1 y δ_2, donde

$$\delta_1 = \min \{c_{ij} - x_{ij} \mid (i,j) \in Y^+ \cup G^+, u_j - u_i = a_{ij}\}$$
$$\delta_2 = \min \{x_{ij} - l_{ij} \mid (i,j) \in Y^- \cup G^-, u_j - u_i = a_{ij}\}$$

El incremento de δ no será mayor a lo necesario para que un arco en mal estado pase a buen estado, si δ es escogido no mayor que δ_3 y δ_4, donde

$$\delta_3 = \min \{ |c_{ij} - x_{ij}| \mid (i,j) \in Y^+ \cup Y^-, u_j - u_i > a_{ij}\}$$
$$\delta_4 = \min \{ |x_{ij} - l_{ij}| \mid (i,j) \in Y^+ \cup Y^-, u_j - u_i < a_{ij}\}$$

Escogimos $\delta = \min \{\delta_1, \delta_2, \delta_3, \delta_4\}$para que no haya incremento en el número de buen estado del arco.

Si δ no es acotada, i.e., si cada una de las deltas δ_i, ($i \leq 4$),es determinada al minimizar un conjunto vacio, entonces no hay una circulación óptima finita. Esto puede ocurrir cuando las capacidades de los arcos en el ciclo son infinitas y el costo neto de la circulación a través del ciclo es negativo.

Supongamos que hay un enlace (S, T) amarillo-rojo, con $s \in S$ y $t \in T$, en el cual todos los arcos amarillos están orientados en la misma dirección que (t, s). Y reorientando todos los arcos a los cuales se les invirtió su dirección al pintarlos de amarillo*.

Un incremento en una pequeña cantidad $\varepsilon > 0$ en el número de nodos de todos los vértices i en T, afecta el valor $u_i - u_i$ solo para los arcos dentro del enlace. Más aun, un cambio así no incrementará el número de buen estado de ningún arco, y tal vez disminuirá el número de buen estado de algunos de los arcos.

Veamos otro ejmplo del siguiente enlace:

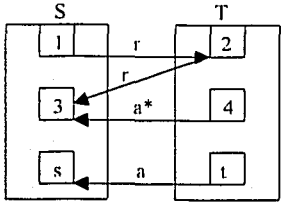


Fig 2.7.6

Reorientando los arcos amarillos*, y aumentando en ε en el número de nodos de los vértices que están en T. Tenemos la siguiente figura 2.7.7, donde la suma de números de estado disminuye.

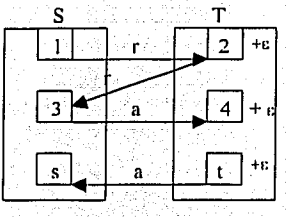


Fig 2.7.7

Veamos el diagrama de buen estado de cada uno de los arcos del enlace de la figura 2.7.7.

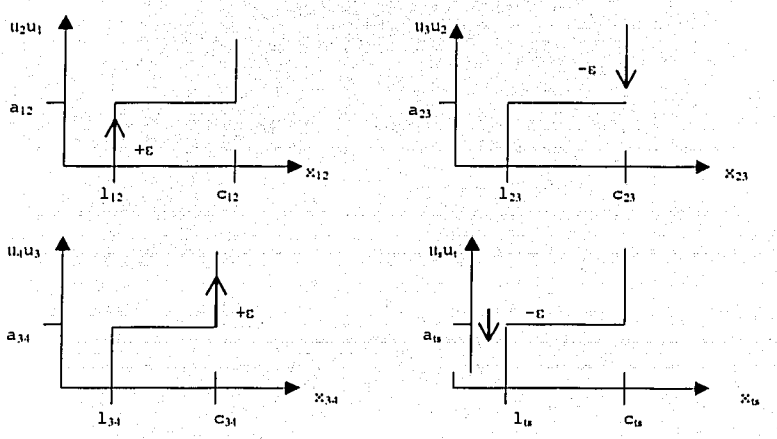


Fig 2.7.8

Ahora veremos el diagrama de buen estado (fig 2.7.9) con todos los cambios posibles de ε en los arcos del enlace amarillo-rojo:

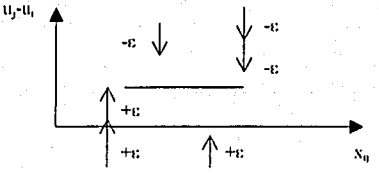


Fig 2.7.9

Como se ve en le diagrama de buen estado de la figura 2.7.9, no hay incremento en el número de buen estado del arco, siempre que ε sea escogida suficientemente pequeña. Enseguida consideraremos como escoger ε.

Para un enlace E amarillo-rojo dado, sean Y y R los subconjuntos de arcos amarillo y rojos del enlace. Igual les asignamos superíndices de + y – para indicar subconjuntos de arcos para los cuales $u_j - u_i$ será incrementado o disminuido en ε respectivamente.

Los arcos en buen estado no pasaran a mal estado si ε no es más grande que ε_1 y ε_2, donde

$$\varepsilon_1 = \{u_j - u_i - a_{ij} \mid (i, j) \in R^+, x_{ij} = c_{ij}\} \equiv$$
$$\varepsilon_2 = \{a_{ij} - u_j + u_i \mid (i, j) \in R^+, x_{ij} = l_{ij}\}$$

El incremento de ε no será mayor al necesario para que el arco en mal estado pase a buen estado si ε es escogida para no ser mayor a ε_3 y ε_4, donde

$$\varepsilon_3 = \{u_j - u_i - a_{ij} \mid (i, j) \in Y^+, l_{ij} \le x_{ij} < c_{ij} \text{ y } u_j - u_i \ne a_{ij}\}$$
$$\varepsilon_4 = \{a_{ij} - u_j + u_i \mid (i, j) \in Y^+, l_{ij} < x_{ij} \le c_{ij} \text{ y } u_j - u_i \ne a_{ij}\}$$

Sea $\varepsilon = \min \{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}$.

Existen tres casos posibles:

Caso1: Si ε no esta acotada, i.e., si cada una de las épsilon, ε_i, ($i \le 4$), es determinada por minimización sobre un conjunto vacío. Esto puede ocurrir solo si $x_{ij} \ge c_{ij}$ para todos los arcos que van de S a T, $x_{ij} \le l_{ij}$ para todos los arcos que van de T a S, y $x_{ts} < l_{ts}$. Como el flujo neto de S a T es cero, entonces:

$$\sum_{i \in S,\, j \in T} l_{ij} > \sum_{i \in S,\, j \in T} c_{ij}$$

Se sigue del teorema de Hoffman que no existe circulación factible.[7]

Caso2: Si ε es finita e igual a ε_3 o ε_4, entonces al menos uno de los arcos en mal estado pasa a buen estado. Ningún número de buen estado es incrementado y probablemente algunos son disminuidos.

Caso3: Si ε es finita y menor que ε_3 y ε_4, ningún arco en mal estado pasa a buen estado. Ningún número de buen estado aumenta y algunos probablemente disminuyen, al menos un arco rojo será pintado de amarillo la próxima vez que sean coloreados. Para tal arco (i, j), si $i \in S$ y $j \in T$, entonces, $l_{ij} = x_{ij} < c_{ij}$, y si $i \in T$ y $j \in S$, entonces $l_{ij} < x_{ij} = c_{ij}$. Además, algunos arcos cambiaran de color de amarillo a rojo. Para cada uno de estos arcos, que $i \in S$ y $j \in T$ implica que $l_{ij} < x_{ij} = c_{ij}$, y que $i \in T$ y $j \in S$ implica que $l_{ij} = x_{ij} < c_{ij}$. Desde luego, ningún arco verde es afectado.

En la implementación del método de transición de estado, que se presenta en éste trabajo, más adelante, se utiliza un procedimiento para etiquetar los arcos, como el de la demostración del teorema de coloración de Minty para construir un ciclo amarillo-verde o un enlace amarillo-rojo.

[7] Dicho teorema se presenta y se demuestra en el apéndice.

El vértice s es etiquetado inicialmente, y todos los demás vértices que se pueden alcanzar desde s son etiquetados sucesivamente. Para usar la analogía de la demostración del teorema de Minty, los arcos verdes se ven como calles de doble sentido, los arcos amarillos como las calles de un solo sentido y los arcos rojos como calles cerradas al transito. Si existe una ruta de s a t, un ciclo amarillo-verde se obtiene tomando las etiquetas en sentido inverso al que se fueron poniendo. Si no existe una ruta de s a t, S contiene todos los vértices etiquetados y T los vértices restantes. El enlace amarillo-rojo es (S, T). (De hecho, (S, T) es un conjunto de corte y no necesariamente un enlace, pero esto es suficiente para nuestros propósitos).

Ahora vamos a probar la convergencia del algoritmo, suponiendo que todas las capacidades mínimas y máximas son enteras, y que el flujo inicial es entero. Posteriormente se verá cómo se mide la complejidad de los algoritmos y cómo se comparan.

Cada vez que se encuentra un ciclo amarillo-verde nuevo nos da una reducción en el número de buen estado de $\delta \geq 1$. Entonces, no más de k revisiones del flujo son necesarias, donde K es la suma de los números de buen estado de la circulación inicial.

Suponiendo que una circulación factible existe, cada vez que se encuentra un enlace amarillo-rojo o un arco que se encontraba en mal estado es llevado a buen estado (caso 2), o al menos un arco rojo cambia de color a amarillo (caso 3). En el primer caso, al menos se reduce un número de buen estado a cero, de tal manera que esto no puede ocurrir más de mín {m, K} veces en total. El segundo caso no puede ocurrir más de (n-1) veces seguidas, por el siguiente razonamiento:

Supongamos que el mismo arco (t, s) se usa para la aplicación del teorema de coloración de Minty hasta que un ciclo amarillo-verde es encontrado. Entonces, cada vez que un enlace es encontrado y ocurre el caso 3, al menos un arco rojo cambia de color a amarillo en siguiente vez que se aplica el procedimiento de poner etiquetas en el caso de que exista una ruta desde s a un vértice adicional i en T. Sigue existiendo una ruta de s a todos los vértices que estaban unidos por una ruta desde s. Cambios de color de amarillo a rojo no tienen consecuencias. Entonces el caso 3 a lo más puede ocurrir (n-1) veces seguidas antes de que se encuentre un ciclo o un arco en mal estado que sea llevado a buen estado (caso 2).

Resumiendo. K es una cota superior del número total de veces que se encuentra un ciclo amarillo-verde o un enlace amarillo-rojo para los cuales el caso 2 se aplique. No puede haber más de (n-1) enlaces amarillo-rojos sucesivos para los cuales el caso 3 se aplica. Por lo tanto, el procedimiento de etiquetar a lo más se aplica (nK) veces en total. Como el procedimiento de etiquetado requiere O(m) unidades de tiempo, y ninguna otra operacion utiliza mas tiempo, se sigue que, O(mnK) es una cota superior del tiempo en que se tarda en correr el algoritmo de transición de estado.

La implementación del algoritmo se hace eficiente utilizando el hecho de que las etiquetas se pueden preservar después de encontrar un enlace para el cual se aplica el caso 3. (Recordando que si existía una ruta que una algún vértice con el vértice s, siguen existiendo rutas que los unen). Esto significa que, en efecto, una aplicación del procedimiento de etiquetar sirve para cada sucesión de enlaces del caso 3 entonces, a lo más k procedimientos completos de etiquetar se requieren, dejando una cota de O(mK).

Utilizaremos dos tipos de etiquetas: permanentes y tentativas. Una etiqueta permanente indica que el vértice que la posee puede ser alcanzado desde s por una ruta amarilla-verde, con todos los arcos amarillos con la misma dirección. Una etiqueta tentativa indica que el vértice será alcanzado por una ruta amarillo-verde una vez que el número de nodos sean revisados por un valor de ε

suficientemente pequeño. Este valor de ε será indicado por un valor π_j asociado con una etiqueta tentativa para el vértice j.

El método de transición de estado fue desarrollado independientemente en 1960 por Minty *ref.(11)* y en 1961 por Fulkerson *ref.(12)*. Aunque en 1959 muchas ideas fueron anticipadas por Yakovleva *ref.(13)*.

2.8 CIRCULACIONES CON CAPACIDADES ENTERAS

El teorema de circulación entera, nos asegura que en el caso de tener capacidades enteras, si existe una solución óptima, ésta es entera. Este resultado es muy importante ya que muchas veces la solución óptima debe ser entera para que podamos ponerla en practica. Por ejemplo, si estamos asignando las personas a trabajos es imposible asignar media persona o un décimo de ésta a un trabajo. Y las matrices unimodulares, que vamos a analizar, son deseables para asegurar una solución óptima entera.

Definición.- Una matriz A es totalmente unimodular si cualquier subdeterminante de ésta es 0, +1 ó −1.

La naturaleza del método de transición de estado es tal que provee una demostración constructiva del teorema de circulación entera, del cual el teorema de flujo entero es un corolario.

> *Teorema.*- Teorema de Circulación Entera.
>
> Si todos las capacidades mínimas y máximas son enteras, y existe una circulación óptima finita, entonces existe una circulación óptima finita entera.

Demostración.- Transformaremos las desigualdades de las capacidades mínimas y máximas en igualdades introduciendo variables de holgura no negativas r_{ij} y s_{ij} de tal manera que:

$$-x_{ij} + r_{ij} = -l_{ij}$$
$$x_{ij} + s_{ij} = c_{ij}.$$

entonces el problema

$$\text{Minimizar } Ax$$

$$\text{sujeto a:} \quad \sum_j x_{ji} - \sum_j x_{ij} = 0 \qquad \forall\, i$$
$$0 \le l_{ij} \le x_{ij} \le c_{ij} \qquad \forall\, i,j$$

ahora esta en la forma:

Minimizar A x

sujeto a: $A(x, r, s) = \bar{b}$

$x, r, s \geq 0,$

donde la matriz A y el vector $\bar{b}$ tienen la siguiente forma:

$$A = \begin{bmatrix} G & 0 & \bar{0} \\ -I_m & I_m & 0 \\ I_m & 0 & I_n \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} 0 \\ -l \\ c \end{bmatrix}$$

Donde G es la matriz de incidencia de la red.

La matriz A, en este caso, es totalmente unimodular (vamos a probarlo).

De la propiedad de unimodularidad se sigue que para cualquier base B, su inversa B^{-1} sólo tiene componentes enteras, y entonces, $x^B = B^{-1}\bar{b}$ sólo tiene componentes enteras si $\bar{b}$ es un vector de componentes enteras. No importa que capacidades mínimas y máximas sean escogidas, toda solución básica factible, incluyendo la solución óptima, es entera.

El siguiente resultado se debe a Hoffman y Kruskal *ref. (14)*

Teorema.- Teorema de Hoffman y Kruskal [8]

En un problema de programación lineal con restricciones $Ax = b$, $x \geq 0$, donde A es una matriz de componentes enteras con renglones linealmente independientes (rango (A) = m), y b es un vector de componentes enteras. Entonces, las 3 condiciones siguientes son equivalentes:

El determinante de cualquier base B es ± 1.

Los puntos extremos del politopo convexo C definido por $Ax = b$, $x \geq 0$ son enteros, para todo vector b de componentes enteras.

La matriz inversa B^{-1} de cualquier base B sólo tiene componentes enteras.

[8] El teorema de Hoffman y Kruskal se demuestra en el apéndice.

Teorema:

Una matriz A(0, +1, -1) es totalmente unimodular si se satisfacen las dos condiciones siguientes:

a) Cada columna de la matriz A contiene a lo más dos elementos diferentes de cero.
b) Los renglones de la matriz A pueden ser divididos en dos conjuntos A_1 y A_2, tal que los elementos de una misma columna diferentes de cero estén en le mismo conjunto de renglones si tienen signos diferentes, y en diferentes conjuntos de renglones si tienen el mismo signo.

Demostración.- Si una matriz A de componentes (0, +1, -1) satisface las condiciones del teorema, entonces, una submatriz de A debe satisfacer las mismas condiciones. ·De aquí, se sigue que es suficiente probar que el det(A) = 0, ± 1 para todas las matrices cuadradas que satisfacen las condiciones. Esto lo demostraremos por la inducción sobre n, donde n es el tamaño de la matriz cuadrada A.

Para cualquier matriz cuadrada A de 1×1, claramente det(A) = 0, ±1.

Supongamos por la hipótesis de inducción que el det(A) = 0, ±1 para toda matriz A de (n-1)×(n-1).

Ahora considere a la matriz A de n×n. Si A contiene una columna de ceros el det(A) = 0. Si alguna columna de A contiene exactamente un elemento diferente de cero, entonces el det(A) = ± det(A') = 0, ±1, donde A' es el cofactor de esa entrada. Si todas las columnas de A contienen exactamente 2 elementos diferentes de cero, entonces

$$\sum_{i \in A_1} a_{ij} = \sum_{i \in A_2} a_{ij} \qquad \text{para } j = 1, 2, \ldots\ldots n$$

Esto implica que det(A) = 0.

Corolario.- Una matriz A(0, +1, -1) es totalmente unimodular si contiene no más de un elemento +1 y no más de un elemento –1 en cada columna.

La matriz de incidencia G es una matriz –(0, +1, -1) con exactamente un elemento +1 y un elemento de –1 en cada columna. Por lo tanto, se sigue inmediatamente del corolario anterior que G es totalmente unimodular.

Teorema:

Una matriz es totalmente unimodular si, y sólo si, cualquier de las matrices A^t, -A, (A,A), (A, I) es totalmente unimodular.

Demostración.- Que A sea totalmente unimodular significa que cualquier subdeterminante de A es +1, -1 o 0.

a. A es totalmente unimodular $\Leftrightarrow$ A^t es totalmente unimodular.

Esto se demuestra utilizando la siguiente propiedad de los determinantes: $\det(A) = \det(A^t)$. De aquí se sigue que cualquier subdeterminante de A^t es +1, -1 o 0.

b. A es totalmente unimodular $\Leftrightarrow$ -A es totalmente unimodular.

Esto se demuestra usando las siguientes propiedades de los determinantes: $\det(...,tA^j,....) = \det(....,A^j,....)$; Si el número de columnas de A es par, entonces el $\det(-A) = \det(A)$, si el numero de columnas de A es impar, entonces, $\det(-A) = -\det(A)$. De aquí se sigue que cualquier subdeterminante de –A es +1, -1 o 0.

c. A es totalmente unimodular $\Leftrightarrow$ (A,A) es totalmente unimodular.

$\Rightarrow$) Suponemos que A es totalmente unimodular. Queremos demostrar que cualquier subdeterminante de (A,A) es 0, +1 o –1 si tomamos cualquier subdeterminante de (A,A), pueden suceder dos cosas. Una, que las columnas de la submatriz sean linealmente dependientes, en cuyo caso el subdeterminante es cero. O que las columnas de la submatriz sean linealmente independientes, en este caso la submatriz de (A,A) también es submatriz de A, y el valor del subdeterminante es 0, +1 o –1.

$\Rightarrow$) Supongamos que (A,A) es totalmente unimodular. A es una submatriz de (A,A), esto implica que $\det(A) = 0, \pm 1$, pero cualquier submatriz de A también es submatriz de (A,A). Por lo tanto, A es una matriz totalmente unimodular.

d. A es totalmente unimodular. Sabemos que la matriz identidad I es totalmente unimodular. Queremos demostrar que cualquier subdeterminante de (A, I) es 0, +1 o –1 si la submatriz de (A, I) es submatriz de A o I, la demostración es obvia. Si no, la submatriz tendrá al menos una columna con solo un elemento diferente de cero (1), y el subdeterminante será el cofactor de este elemento y así, obtendremos que el subdeterminante de (A, I) es solo un subdeterminante de la matriz A. Por lo tanto, la matriz (A, I) es totalmente unimodular.

$\Rightarrow$) Supongamos que (A, I) es totalmente unimodular. A es una submatriz de (A, I), esto implica que $\det(A) = 0, \pm 1$, pero cualquier submatriz de A también es submatriz de (A, I). Por lo tanto, A es una matriz totalmente unimodular.

De aquí vemos que utilizando el teorema anterior para cualquier secuencia de transformaciones posibles de la matriz

$$A = \begin{bmatrix} G & I_n & 0 & 0 \\ -I_m & 0 & I_m & 0 \\ I_m & 0 & 0 & I_m \end{bmatrix}$$

La matriz A es totalmente unimodular. La matriz A del teorema 1 es una submatriz de esta matriz A, y por lo tanto, hemos establecido el resultado deseado.

Un problema de programación lineal con una matriz de coeficientes totalmente unimodulares nos da una solución óptima entera para cualquier vector objetivo y cualquier vector de elementos enteros del lado derecho de las restricciones.

Primeramente veremos cómo se construye el programa: qué hace, cuáles son las variables de entrada y de salida, así como las opciones de interacción que se pueden tener. El codigo del programa se presenta en el anexo. Para la implementación del algoritmo se utilizo el lenguaje de programación Turbo Pascal versión 7.0.

ALGORITMO DE TRANSICIÓN DE ESTADO.[9]

0. Verificar que todos los arcos del grafo cumplan con la ley de conservación.

   Si cumple, entonces, dar una solución inicial primal y dual x = (0) y u = (0).
   Asignar a todas las etiquetas π_i el valor de infinito.

1. Buscar arcos en mal estado.

   Si se encuentran, entonces, colorear los arcos, según el teorema de Minty.
   Tomar un arco(T, S) y asignar al nodo S. etiqueta permanente.
   En caso de no encontrar arco en mal estado, entonces terminar el algoritmo.

1.1. Si existen nodos sin examinar de etiqueta permanente, hacer:

   Recorrer todos los arcos (i, j)
   Si el arco(i, j) es de color amarillo o verde y verde(j, i), entonces, asignar etiqueta permanente al nodo j

1.2 En caso de ser una arco(i, j) de color rojo. Si $x_{ij} = l_{ij}$ y $a_{ij} - (u_j - u_i) < \pi_i$, entonces, al nodo j se le asigna etiqueta permanente.

   En caso que el arco(j, i) de color rojo, cumple con $x_{ij} = c_{ij}$ y $u_i - u_j - a_{ij} < \pi_i$, al nodo j se le asigna etiqueta tentativa y $\pi_i = u_i - u_j - a_{ij}$

1.3 Verificar si el nodo T tiene etiqueta permanente.

[9] Las ideas del algoritmo de transición de estado fueron tomadas del trabajo "Circulación con costo mínimo" E. Brito, 1989.

Si es falso, entonces, repetir los pasos anteriores desde 1.1, hasta que T tenga etiqueta permanente.

En otro caso:

2. Buscar un ciclo amarillo-verde con etiqueta permanente en sentido T, S.

2.1 Si δ no es acotada, entonces, finalizar algoritmo.

En otro caso:

2.2 Incrementar o disminuir una mínima cantidad $\delta > 0$ a la capacidad de cada arco del ciclo.

Verificar si cumple con la ley de la conservación.
Si es cierto, entonces, borrar todas las etiquetas de los nodos e ir al paso 1.0
En otro caso: Finalizar.

En otro caso:

3. Encontrar un enlace (S, T).

Comprobar si existe el enlace mediante el teorema de Hoffman.
Si no existe circulación factible, entonces: terminar.

En otro caso:

Determinar ε.
Si ε es acotada, entonces, $u_i = u_i + \varepsilon$
Si ε es finito e igual a ε_i, entonces, ir al paso 1.0
Si ε es finito y menor a ε_i, entonces:
$\pi_i = \pi_i - \varepsilon$
Asignar etiqueta permanente a todos los nodos con $\pi_i = 0$.
Ir al paso1.1

Fin del algoritmo.[10]

[10] La sección 2.7 (metodo de transición de estado) contiene la explicación sobre cómo se colorean los arcos, cómo se calculan las δ y ε, y se definen los casos 2 y 3 del paso 3.

CAPÍTULO III

ALGORITMOS POLINOMIALES PARA EL PROBLEMA DE FLUJO A COSTO MÍNIMO

En este capitulo se verá la forma de analizar y comparar la complejidad de los algoritmos, principalmente los algoritmos con cotas polinomiales. La técnica de escalamiento de Edmonds y Karp, y como se aplica al método de transición de estado. Y finalmente, se compararán algunos de los algoritmos más importantes que resuelven el problema de circulación con costo minimo.

3.1 ANÁLISIS DE LA COMPLEJIDAD DE LOS ALGORITMOS

La forma más aceptada de medir la complejidad de un algoritmo es el tiempo que se necesita para que produzca el resultado final. Este periodo de tiempo puede variar demasiado de una computadora a otra debido a su velocidad y al conjunto de instrucciones. Para evitar esto y poder comparar los algoritmos de una manera eficaz, se expresa el tiempo que requiere el algoritmo para su ejecución en una computadora hipotética en términos de pasos elementales como: operaciones aritméticas, comparaciones, asignaciones, etc. La computadora hipotética tiene memoria ilimitada y realizar cada paso elemental toma únicamente una unidad de tiempo.

Como el número de pasos requeridos por un algoritmo no es el mismo para todos los datos de entrada; se considera un mismo tamaño n dado; y se define la complejidad del algoritmo para ese tamaño n en el peor de los casos para cualquiera de estos datos de entrada. Así , la complejidad del algoritmo es una función del tamaño de los datos de entrada, como $10n^3$, 2^n, y $n\log_{10}n$.

Muchas veces en el estudio de la complejidad de un algoritmo, sólo nos interesa el comportamiento del algoritmo cuando el número de datos de entrada es muy grande, por que esto es lo que va a determinar los límites de la aplicabilidad del algoritmo. Diferencias entre un algoritmo de complejidad $10n^3$ y otro de complejidad $9n^3$ pueden hacerse irrelevantes por un avance tecnológico que incremente la velocidad de las computadoras. También los términos que crecen muy lentamente (como el término 5n en $n\log_2 n + 5n$) van a ser dominados por los términos que crecen más rápido para n suficientemente grande (en el ejemplo $n>1000$). Es por esto que nos interesa la tasa de crecimiento de la complejidad del algoritmo.

Definición - Sean $f(n)$: $\mathbb{Z}^+ \rightarrow \mathbb{R}^+$ y $g(n)$: $\mathbb{Z}^+ \rightarrow \mathbb{R}^+$ dos funciones. Escribimos $f(n) = O(g(n))$ si existe una constante $c>0$ tal que para una n suficientemente grande, $f(n) \leq cg(n)$.

Usando esta notación, la tasa de crecimiento de la complejidad de un algoritmo puede ser acotada por frases como esta: "toma $O(n^3)$ unidades de tiempo".

Pero la pregunta que podemos tener es: ¿Cómo debemos considerar que un problema computacional es resuelto satisfactoriamente? La respuesta toma en consideración los algoritmos que resuelven este problema. El primer criterio es que exista uno para este problema que no necesita mucho tiempo para resolverlo; en este caso el problema se considera resuelto, y no de otra manera. De hecho, como ya mencionamos anteriormente, es la tasa de crecimiento de la mejor cota del tiempo la que va a determinar la utilidad práctica del algoritmo. Entonces, la siguiente pregunta que podamos tener es: ¿qué tasas de crecimiento debemos considerar como soluciones aceptables para los problemas computacionales?

Hoy en día hay un acuerdo general de que un algoritmo es una solución practica útil para un problema computacional sólo si su complejidad crece polinomialmente con respecto al tamaño de los datos de entrada. Los algoritmos de complejidad $O(n)$ o $O(n^3)$ son aceptables en esta escuela de pensamiento. (la tasa de crecimiento de forma polinomial de un algoritmo esta especificada por su

grado). Naturalmente, los algoritmos cuya complejidad no esta acotada asintoticamente por una cota polinomial, pero están acotados por un polinomio los que también son considerados como soluciones aceptables útiles.

Para entender la importancia de la clase de algoritmos acotados por cotas polinomiales debemos considerar los algoritmos que no son acotados por ningún polinomio para números de datos de entrada lo suficientemente grandes. Usualmente nos referimos a estos algoritmos como exponenciales, porque 2^n es el paradigma de tasas de crecimiento no polinomiales. Algunas tasas de crecimiento exponenciales son: k^n para cualquier k>1 fija, n!, 2^{n3}, n^n y n^{log}. Es obvio que cuando el tamaño de los datos de entrada crece, cualquier algoritmo polinomial será mas eficiente que cualquier algoritmo exponencial.

En la tabla 3 se ve la tasa de crecimiento de funciones polinomiales y exponenciales.

Función	Valores aproximados		
n	10	100	1000
nlogn	33	664	9966
n^3	1000	1000000	10^9
10^6n^8	10^{14}	10^{22}	10^{30}
2^n	1024	$1.27*10^{30}$	$1.05*10^{301}$
n^{logn}	2099	$1.93*10^{13}$	$7.89*10^{29}$
n!	3628800	10^{158}	$4*10^{2567}$

Tabla 3

Otra característica positiva de los algoritmos polinomiales es que, en un sentido, pueden obtener mayor beneficio de los adelantos tecnológicos. Cada vez que un avance tecnológico aumenta la velocidad de las computadoras diez veces, el tamaño el tamaño del problema mas grande que puede resolver un algoritmo polinomial en una hora, será multiplicado por una constante entre 1 y 10. en contraste, un algoritmo exponencial solo experimentara un incremento aditivo en el tamaño del problema que puede resolver en un determinado tiempo.

En la siguiente tabla 4 se ve como los algoritmos exponenciales obtienen un mayor beneficio de los adelantos tecnológicos.

Función	Tamaño del ejemplo resuelto en un día	
	En una computadora	En una computadora 10 veces más rápida
n	10^{12}	10^{13}
nlogn	$0.948*10^{11}$	$0.87*10^{12}$
n^2	10^6	$3.16*10^6$
n^3	10^4	$2.15*10^4$
10^8n^4	10	18
2^n	40	43

Función	Tamaño del ejemplo resuelto en un día	
	En una computadora	En una computadora 10 veces más rápida
$n^{\log n}$	79	95
n!	14	15
10^n	12	13

Tabla 4

Tenemos que mencionar también que los algoritmos polinomiales tienen propiedades útiles: se pueden combinar para resolver casos especiales del mismo problema; un algoritmo polinomial puede invocar a otro algoritmo polinomial como una subrutina y el algoritmo resultante seguirá siendo polinomial.

3.2 TECNICA DE ESCALAMIENTO DE EDMOND Y KARP

El dicho mejoramiento técnico en la eficiencia del método de transición de estado se realizará aplicando la técnica de escalamiento de Edmonds y Karps.

A partir del análisis del método de transición de estado que hicimos en el capítulo dos, la cota del orden O(Km) se puede establecerse para el número de pasos que requiere el método, donde K es la suma de los números de buen estado de todos los arcos de las soluciones primal y dual iniciales. Si tomamos como soluciones iniciales x = (0) y u = (0), entonces, K será tan grande como la suma de las capacidades de todos los arco, las cuales asumimos enteras.

Para que se pueda decir que el método de transición de estado es un algoritmo eficiente, el número de pasos requeridos por este método debe tener una cota de orden polinomial en el logaritmo de las magnitudes de las capacidades de los arcos, que es el número de bits requeridos para especificarlas como datos de entrada en la computadora.

En la técnica de escalamiento desarrollada en 1972 por Edmonds y Karp *ref (15)*, que veremos a continuación, el algoritmo de transición de estado se aplica a una serie de problemas, los cuales dan aproximaciones sucesivamente más cercanas a la solución del problema original. Y se obtiene una cota del origen polinomial buscado.

Si queremos aplicar el método de transición de estado a un problema con cotas inferiores y capacidades enteras, y para el cual la capacidad máxima no es mayor que 2^p, donde el problema original es:

$$p = [\log_2 c_{ij}], \text{ para la mayor de las } c_{ij}.$$

tenemos que hacer lo siguiente:

Primero reemplazamos el problema original por el problema (0)[11] en el cual:

$$c_{ij}^{(0)} = [c_{ij}/2^p]$$
$$l_{ij}^{(0)} = [l_{ij}/2^p]$$

y el costo a_{ij} se toma como fue dado en el problema original. Todas las cotas inferiores y capacidades son 0 ó 1. Esta aproximación de orden – 0 de la red original admite una circulación factible si existe una circulación factible en el problema original.

Hay que notar que:

$$2^p c_{ij}^{(0)} \geq c_{ij}$$
$$2^p l_{ij}^{(0)} \leq l_{ij}.$$

Si tomamos $u = (0)$ y $x = (0)$ como un flujo inicial, en esta aproximación del problema original, todos los números de buen estado son 0 ó 1, y $K \leq m$, donde m es el número de arcos. Por lo tanto, el método de descomposición no requiere más de $O(m^2)$ pasos para obtener las soluciones primal y dual optimas $x^{(0)}$ y $u^{(0)}$.

Ahora consideremos el problema (1) en el cual

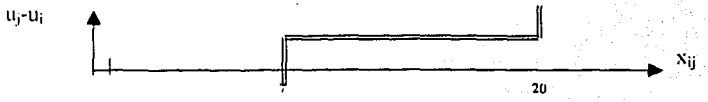
$$c_{ij}^{(1)} = [c_{ij}/2^{p-1}]$$
$$l_{ij}^{(1)} = [l_{ij}/2^{p-1}]$$

Las cotas a_{ij} se quedan como fueron dadas. Ahora todas las cotas inferiores y capacidades son 0, 1 ó 2. Si tomamos $2x^{(0)}$, $u^{(0)}$ como soluciones primal y dual inicial respectivamente, todos los números de buen estado otra vez son 0 ó 1 y $K \leq m$. El método de transición de estado no necesita mas de O(m2) pasos para llegar a las soluciones primal y dual optimas $x^{(1)}$ y $u^{(1)}$

Generalizando y pasando del problema (k) al problema (k+1), tomando $2x^{(k)}$, $u^{(k)}$ como soluciones iniciales para el problema (k+1), llegamos al problema (p), el cual resuelve el problema de una red idéntica a la del problema original. De esta manera obtenemos un flujo para el en $O(pm^2)$ pasos. Así obtenemos el resultado deseado para el problema original.

Ejemplo.- Vamos a ver los diagramas de buen estado para un arco con cota inferior $l_{ij} = 7$ y capacidad $c_{ij} = 20$, aplicando la técnica de escalamiento.

Diagrama de buen estado del problema original es:

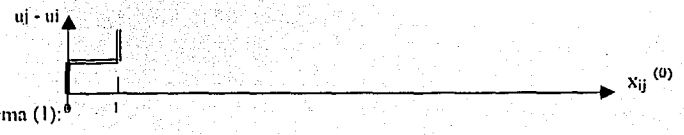


[11] El número entre parientesis representa el indice de la iteración. Por lo general, se empieza con el indice 0 la solución del problema.

$p = [\log_2 c_{ij}] = [\log_2 20] = [4.32] = 5$

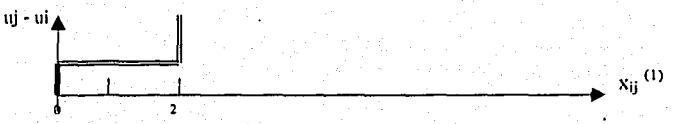
Problema (0):

$$c_{ij}^{(0)} = [c_{ij}/2^p] = [20/2^5] = [0.625] = 1$$
$$l_{ij}^{(0)} = [l_{ij}/2^p] = [7/2^5] = [0.21875] = 0$$



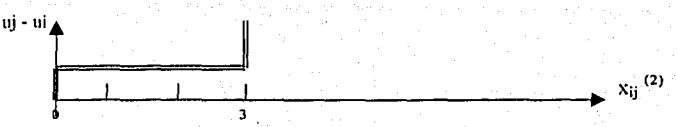
Problema (1):

$$c_{ij}^{(1)} = [c_{ij}/2^{p-1}] = [20/2^4] = [1.25] = 2$$
$$l_{ij}^{(1)} = [l_{ij}/2^{p-1}] = [7/2^4] = [0.4375] = 0$$



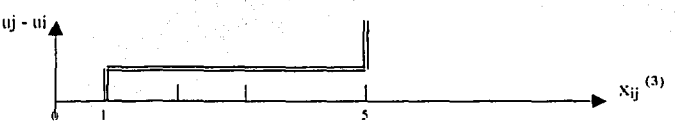
Problema (2):

$$c_{ij}^{(2)} = [c_{ij}/2^{p-2}] = [20/2^3] = [2.5] = 3$$
$$l_{ij}^{(2)} = [l_{ij}/2^{p-2}] = [7/2^3] = [0.875] = 0$$



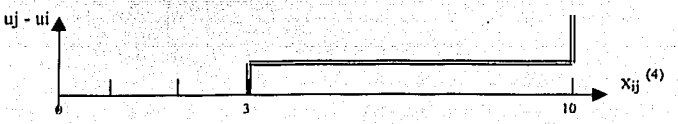
Problema (3):

$$c_{ij}^{(3)} = [c_{ij}/2^{p-3}] = [20/2^2] = [5] = 5$$
$$l_{ij}^{(3)} = [l_{ij}/2^{p-3}] = [7/2^2] = [1.75] = 1$$



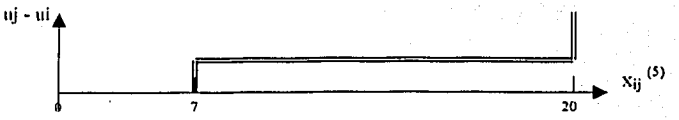
Problema (4):

$$c_{ij}^{(4)} = [c_{ij}/2^{p-4}] = [20/2^1] = [10] = 10$$
$$l_{ij}^{(4)} = [l_{ij}/2^{p-4}] = [7/2^1] = [3.5] = 3$$



Problema (5):

$$c_{ij}^{(5)} = [c_{ij}/2^{p-5}] = [20/2^0] = [20] = 20$$
$$l_{ij}^{(5)} = [l_{ij}/2^{p-5}] = [7/2^0] = [7] = 7$$



Este diagrama de buen estado es el mismo al diagrama de problema original. Por lo tanto, después de realizar la prueba podemos concluir que la técnica de escalamiento, aunque es bastante fácil de implementar, probablemente es de importancia practica limitada.

Todos los algoritmos para resolver las redes de flujo a costo mínimo de tiempo polinomial se basan en la técnica de escalamiento. Fueron Edmonds y Karp quienes desarrollaron el primer algoritmo polinomial para este tipo de problemas usando escalamiento en las capacidades. La técnica de escalamiento también ha sido utilizada para otro tipo de problemas de optimización en redes. Esta técnica puede ser aplicada ya sea en el costo (escalamiento de costo) o en las capacidades mínima y máxima (escalamiento de las capacidades). Róck fue quien propuso la utilización del método de escalamiento en el costo.

En la siguiente tabla 5 se presentan los algoritmos polinomiales conocidos para el problema de flujo a costo mínimo.

Fecha	Desarrollados por	ref	Tiempo en el que corre
1972	Edmonds and Karp	*(16)*	O(mlog (U)S(n, m, C))
1980	Róck	*(17)*	O(mlog (U)S(n, m, C))
1984	Orlin	*(18)*	$O(m^2 \log (n) S(n, m, C))$
1985	Fujishige	*(19)*	$O(m^2 \log (n) S(n, m, C))$
1986	Galil y Tardos	*(20)*	$O(n^2 \log (n) S(n, m, C))$
1988	Orlin	*(21)*	O(mlog (n)S(n, m, C))
1980	Rock	*(17)*	O(mlog (C)F(n, m, U))
1984	Tardos	*(22)*	$O(m^4)$
1985	Bland y Jensen	*(23)*	O(nlog (C)F(n, m, U))
1987	Goldberg y Tarjan	*(24)*	O(nmlog(n²/m)min {log(nC), mlogn}
1987	Goldberg y Tarjan	*(24)*	O(nB(n, m) min {log (nC), mlog(n))
1987	Goldberg y Tarjan	*(23)*	O(nmlog (n)min{log(nC), mlog(n)))

Tabla 5

Donde las cotas son dadas en términos de las siguientes funciones y parámetros:

n	Número de vértices
m	Número de arcos
U	Máximo valor absoluto de la capacidad de un arco (se asume que es un número entero).
C	Máximo valor absoluto del costo de un arco (se asume que es un número entero).
S(n, m, C)	Tiempo en que se resuelve el problema de la ruta más corta con sólo un nodo fuente y con costos de arcos no negativos, como una función de n, m y C.
F(n, m, U)	Tiempo en que se encuentra un flujo máximo, como una función de n, m y U.
B(n, m)	Tiempo en que se encuentra el flujo de bloqueo en una red acíclica, como una función de n y m.

Las mejores cotas establecidas para S, F y B son:

$S(n, m, C) = O(n\log(n) + m)$
$= O(m\log(\log(C)))$
$= O(n(\log (C))^{1/2} + m)$

$F(n, m, U) = O(nm\log (n^2/m))$
$= O(nm\log (n/m) (\log(U))^{1/2} + 2)$

$B(n, m) = O(m\log (n^2/m))$

Como podemos apreciar los algoritmos de la tabla anterior están separadas en 3 grupos:

1.- Los que usan la técnica de escalamiento de las capacidades.
2.- Los que aplican el escalamiento del costo.
3.- Algoritmos que solo usan la técnica de escalamiento es su análisis.

Todos los algoritmos que usan el escalamiento en las capacidades necesitan una subrutina que encuentre la ruta más corta; todos los algoritmos que usan el escalamiento en el costo requieren de una subrutina que calcule el flujo máximo o similar.

Los algoritmos de Tardos, Orlin (1984 y 1988), Fujishige, Galil y Tardos, y los dos algoritmos de Goldberg y Tarjan que utilizan la técnica de escalamiento en su implementación, tiene un comportamiento polinomial muy marcado: corren en un tiempo polinomial tanto en n, como en m asumiendo que las operaciones aritméticas requieran de O(1) unidades de tiempo, y los valores que manipulan tienen un numero de bits polinomial en n, m, log(U) y log (C).

Comparar las velocidades relativas en que corren varios algoritmos es algo complicado debido a que sus cotas dependen de diferentes formas de n, m, U yC.

Entre los algoritmos que se conocian antes de que Goldberg y Tarjan desarrollaran los suyos, los algoritmos dominantes eran los de Edmonds y Karp, Röck, Bland y Jensen, y el de Galil y Tardos. Bajo la suposición de que log(U) y log(C) son Θ (logn) los algoritmos de Edmonds y Karp y el de Röck obtienen la mejor cota de $O(m^2\log(n) + nm(\log(n))^2)$. El algoritmo de Orlin del año 1988, tiene la misma cota pero su ventaja esta en que no utiliza la suposición anterior.

CAPÍTULO IV

APLICACIONES

En el capitulo cuatro se describe cómo se puede aplicar el método de transición de estado a problemas reales, como son el problema de plantación de proyectos, el de transporte y el de transbordo, como al problema de asignación.

Una de las aplicaciones de la teoría de flujo de redes más útil es la de planeación de proyectos. Varias técnicas han sido desarrolladas como: *CPM* (Critical Path Meted – Método de la Ruta Crítica) para resolver problemas de duración predeterminada; y *PERT* (Project Evaluation and Review Technique – Técnica de Evaluación y Revisión del Programa) para problemas de duración probabilística o estocástica.

Si tenemos un proyecto grande, supongamos que este puede ser dividido en un número determinado de tareas. Las relaciones de precedencia entre éstas se indican asociando las tareas con los arcos de una grafica dirigida. Todas las tareas dirigidas hacia un vértice deben de ser completadas antes de que cualquier tarea que salga de este vértice sea empezada. (habrá que introducir variables ficticias que tengan un tiempo de realización cero, de manera que sea posible adecuar al modelo todas las relaciones de precedencia de un conjunto dado de tareas).

Asociando con cada tarea (i, j) están su tiempo de realización "normal" a_{ij}, su tiempo de realización de "urgencia" b_{ij}, y el costo c_{ij} de disminuir la tarea en una unidad de tiempo (se cree que haya la aplicación de tiempo extra o de una fuerza de trabajo mayor). Esto es, si t_{ij} es la duración actual de la tarea, entonces, $b_{ij} \le t_{ij} \le a_{ij}$, y el costo requerido para completar la tarea en este tiempo t_{ij} es $c_{ij}(a_{ij} - t_{ij})$

A cada vértice i de la red del proyecto le asociamos una variable u_i, donde u_i denota el tiempo en el cual el "evento" i ocurra. Si el vértice s es el evento inicial del proyecto, y el vértice t el evento final. Entonces, el problema de encontrar el costo mínimo de reducir el proyecto a una duración dada T es:

$$\text{Minimizar } z = \sum_{i,j} c_{ij}(a_{ij} - t_{ij})$$

como $z = \sum_{i,j} c_{ij}\, a_{ij} - \sum_{i,j} c_{ij}\, t_{ij}$ y $c_{ij}\, a_{ij}$ es constante, entonces minimizar z es equivalente a:

$$\text{Maximizar } \sum_{i,j} c_{ij}\, t_{ij}$$

sujeto a:
$$u_t - u_s \le T$$
$$u_i - u_j + t_{ij} \le 0$$
$$b_{ij} \le t_{ij} \le a_{ij}$$
u_i no restringida.

Para obtener el modelo dual de este problema de programación lineal vamos a introducir las variables no negativas v, x_{ij}, α_{ij} y β_{ij} (que serán las variables ficticias o de holgura del modelo) asociadas respectivamente con las restricciones $u_t - u_s \le T$, $u_i - u_j + t_{ij} \le 0$, $t_{ij} \le a_{ij}$ y $-t_{ij} \le -b_{ij}$.

Al realizar las modificaciones, obtenemos la forma dual del problema, el cual es el siguiente:

$$\text{Minimizar } w = \sum_{i,j} a_{ij}\,\alpha_{ij} - \sum_{i,j} b_{ij}\,\beta_{ij} + Tv$$

$$\text{sujeto a:} \quad \sum_{j} x_{ji} - \sum_{j} x_{ij} = \begin{cases} -v & \text{si } i = s \\ 0 & \text{si } i \neq s,t \\ v & \text{si } i = t \end{cases} \qquad (1)$$

$$x_{ij} + \alpha_{ij} - \beta_{ij} = c_{ij}$$
$$x_{ij},\ \alpha_{ij},\ \beta_{ij} \geq 0$$

Vamos a ver un ejemplo de un problema de planeación de proyectos.

Ejemplo.- Una compañía desea introducir un producto nuevo y necesita planear las diferentes tareas a seguir, para que el proyecto sea finalizado en 85 días. Los datos de las tareas se muestran en la tabla 6. (los tiempos están en días y los costos en millones de pesos).

Tarea	Tarea inmediata anterior	Tiempo normal (a_{ij})	Tiempo de urgencia (b_{ij})	Costo (c_{ij})
A	----------	10	10	----------
B	A	25	20	75
C	A	20	18	100
D	A	30	28	30
E	B	20	17	80
F	C	20	15	23
G	D, F	10	8	2
H	E, G	30	26	94

Tabla 6

Gráficamente el proyecto se representa como se ve en la figura 4.1.1:

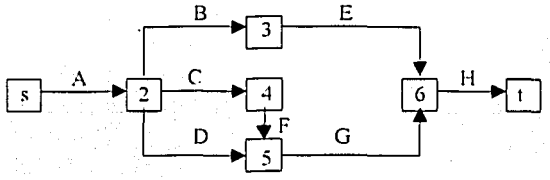


Fig 4.1.1

Expresando esta problema como un problema de programación lineal, queda:

$$\text{Maximizar } z = 75t_{23} + 100t_{24} + 30t_{25} + 80t_{36} + 23t_{45} + 2t_{56} + 94t_{61}$$

sujeto a:

$u_1 - u_s \le 85$	
$u_s - u_2 + t_{s2} \le 0$	$10 \le t_{s2} \le 10$
$u_2 - u_3 + t_{23} \le 0$	$20 \le t_{23} \le 25$
$u_2 - u_4 + t_{24} \le 0$	$18 \le t_{24} \le 20$
$u_2 - u_5 + t_{25} \le 0$	$28 \le t_{25} \le 30$
$u_3 - u_6 + t_{36} \le 0$	$17 \le t_{36} \le 20$
$u_4 - u_5 + t_{45} \le 0$	$15 \le t_{45} \le 20$
$u_5 - u_6 + t_{56} \le 0$	$8 \le t_{56} \le 10$
$u_6 - u_1 + t_{61} \le 0$	$25 \le t_{61} \le 30$

Y las matrices asociadas son:

$$A = \left(\begin{array}{cccccccc|ccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{array}\right)$$

$$\bar{b} = \begin{pmatrix} 10 \\ 25 \\ 20 \\ 30 \\ 20 \\ 20 \\ 10 \\ 30 \\ -10 \\ -20 \\ -18 \\ -28 \\ -17 \\ -15 \\ -8 \\ -26 \\ 85 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \bar{f} = \begin{pmatrix} t_{a2} \\ t_{23} \\ t_{24} \\ t_{25} \\ t_{36} \\ t_{45} \\ t_{56} \\ t_{6t} \\ u_a \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_t \end{pmatrix} \qquad \bar{c} = \begin{pmatrix} 0 \\ 75 \\ 100 \\ 30 \\ 80 \\ 23 \\ 2 \\ 94 \end{pmatrix}$$

Si tomamos el vector $\bar{u}$ de esta forma

$$\bar{u} = \begin{pmatrix} \alpha_{a2} \\ \alpha_{23} \\ \alpha_{24} \\ \alpha_{25} \\ \alpha_{36} \\ \alpha_{45} \\ \alpha_{56} \\ \alpha_{6t} \\ \beta_{a2} \\ \beta_{23} \\ \beta_{24} \\ \beta_{25} \\ \beta_{36} \\ \beta_{45} \\ \beta_{56} \\ \beta_{6t} \\ v \\ x_{a2} \\ x_{23} \\ x_{24} \\ x_{25} \\ x_{36} \\ x_{45} \\ x_{56} \\ x_{6t} \end{pmatrix}$$

Y ahora multiplicando la matriz A^T por π, obtenemos:

$$A^T \pi = \begin{pmatrix} \alpha_{s2} - \beta_{s2} + x_{s2} \\ \alpha_{23} - \beta_{23} + x_{23} \\ \alpha_{24} - \beta_{24} + x_{24} \\ \alpha_{25} - \beta_{25} + x_{25} \\ \alpha_{36} - \beta_{36} + x_{36} \\ \alpha_{45} - \beta_{45} + x_{45} \\ \alpha_{56} - \beta_{56} + x_{56} \\ \alpha_{6t} - \beta_{6t} + x_{6t} \\ -v + x_{s2} \\ -x_{s2} + x_{23} + x_{24} + x_{25} \\ -x_{23} + x_{36} \\ -x_{24} + x_{45} \\ -x_{25} - x_{45} + x_{56} \\ -x_{36} - x_{56} + x_{6t} \\ v - x_{6t} \end{pmatrix} = \begin{pmatrix} 0 \\ 75 \\ 100 \\ 30 \\ 80 \\ 23 \\ 2 \\ 94 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = c$$

De aquí tenemos que la representación de este problema de plantación en su forma dual será:

$$\text{Minimizar } w = 10\alpha_{s2} + 25\alpha_{23} + 20\alpha_{24} + 30\alpha_{25} + 20\alpha_{36} + 20\alpha_{45} + 10\alpha_{56} + 30\alpha_{6t} - 10\beta_{s2} - 20\beta_{23} - 18\beta_{24} - 28\beta_{25} - 17\beta_{36} - 15\beta_{45} - 8\beta_{56} - 26\beta_{6t} - 85v$$

sujeto a:

$$\begin{aligned} &\alpha_{s2} - \beta_{s2} + x_{s2} = 0 && -x_{s2} = -v \\ &\alpha_{23} - \beta_{23} + x_{23} = 75 && -x_{s2} + x_{23} + x_{24} + x_{25} = 0 \\ &\alpha_{24} - \beta_{24} + x_{24} = 100 && -x_{23} + x_{36} = 0 \\ &\alpha_{25} - \beta_{25} + x_{25} = 30 && -x_{24} + x_{45} = 0 \\ &\alpha_{36} - \beta_{36} + x_{36} = 80 && -x_{25} - x_{45} + x_{56} = 0 \\ &\alpha_{45} - \beta_{45} + x_{45} = 23 && -x_{36} - x_{56} + x_{6t} = 0 \\ &\alpha_{56} - \beta_{56} + x_{56} = 2 && x_{6t} = v \\ &\alpha_{6t} - \beta_{6t} + x_{6t} = 94 \end{aligned}$$

$$x_{ij}, \alpha_{ij}, \beta_{ij} \geq 0$$

De $a_{ij} \geq b_{ij} \geq 0$ y (1)[12], se ve que la solución óptima debe satisfacer las soluciones siguientes:

$$x_{ij} \leq c_{ij} \quad \Rightarrow \quad \alpha_{ij} = c_{ij} - x_{ij}, \quad \beta_{ij} = 0$$

y

$$x_{ij} > c_{ij} \quad \Rightarrow \quad \alpha_{ij} = 0, \quad \beta_{ij} = x_{ij} - c_{ij}$$

[12] El modelo dual de la pagina 110.

De acuerdo con esto, sustituyendo α_{ij} y β_{ij} en el problema dual, un problema de flujo equivalente es:

$$\text{Minimizar } z = \sum T_{ij}(x_{ij}) + Tv$$

$$\text{sujeto a: } \sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v & \text{si } i = s \\ 0 & \text{si } i \neq s,t \\ v & \text{si } i = t \end{cases}$$

$$x_{ij} \geq 0, \text{ donde}$$

$$T_{ij}(x_{ij}) = \begin{cases} a_{ij}(c_{ij} - x_{ij}) & \text{si } x_{ij} \leq c_{ij} \\ b_{ij}(c_{ij} - x_{ij}) & \text{si } xij > c_{ij} \end{cases}$$

Y $T_{ij}(x_{ij})$ tiene la forma, como se ve en la siguiente figura:

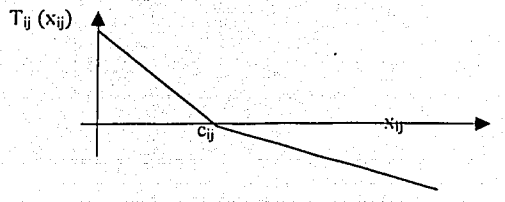


Fig 4.1.

La convexidad de $T_{ij}(x_{ij})$ se debe al hecho de que $a_{ij} > b_{ij}$.

Podemos ver cada tarea (i, j) del proyecto representándola por dos arcos paralelos del vértice i al j en la red de flujo, uno con costo $-a_{ij}$ y capacidad c_{ij} y el otro con costo $-b_{ij}$ y capacidad infinita. Esto se sigue del hecho de que

$$\text{Minimizar } z = \sum T_{ij}(x_{ij}) + Tv$$

Es equivalente a:

$$\text{Minimizar} \begin{cases} \sum_{i,j} a_{ij}(c_{ij} - x_{ij}) + Tv & \text{si } x_{ij} \leq c_{ij} \\ \sum_{i,j} b_{ij}(c_{ij} - x_{ij}) + Tv & \text{si } x_{ij} > c_{ij} \end{cases}$$

Si existen ciclos dirigidos en la red del proyecto el problema de flujo no tendrá una solución óptima finita para ningún valor de flujo v. Pero hay que notar que las relaciones de precedencia de las tareas son tales que la red es necesariamente acíclica.

Podemos añadir al flujo de la red un arco (t, s) con capacidad infinita y costo T, el flujo a través de este arco es v. Para cualquier T especificada, el problema de circulación puede ser resuelto por el método de transición de estado.

El diagrama de estado para un arco (i, j) típico es el siguiente (fig 4.1.2):

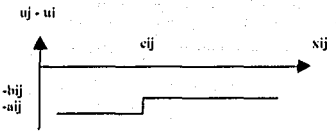


Fig 4.1.2

Podemos variar el parámetro T y observar la circulación óptima que resulta. Si T es escogida lo suficientemente grande, esperamos que la circulación cero sea óptima, reflejando el hecho de que si la duración del proyecto es lo suficientemente grande, ningún dinero se gastará para reducir el tiempo de las tareas. Esto será sierto para cualquier T tan grande como la ruta crítica, o la más larga ruta de s a t con respecto a las longitudes a_{ij} de los arcos.

Definición.- Una ruta crítica es un conjunto de tareas sucesivas, que no pueden sufrir un retraso sin aumentar el tiempo que lleva finalizar el proyecto.

Por otra parte, si T se escoge lo suficientemente pequeña, se espera que no exista una circulación óptima finita, correspondiente al hecho de que ningún gasto finito de dinero puede reducir la duración del proyecto más allá de un cierto punto. Este será el caso para cualquier valor T menor que la longitud de la ruta más larga de s a t con respecto a las longitudes b_{ij} de los arcos.

Realizando el análisis paramétrico al resolver el problema de la ruta más larga con respecto a las longitudes a_{ij} de los arcos, los números de nodos u_i determinados junto con la circulación cero proveen soluciones óptimas primal y dual para $T \geq u_t$. Entonces, el parámetro T se reduce. Todos los arcos permanecen en buen estado con excepción del arco (t, s). El método de transición de estado se aplica para regresar al arco (t, s) a buen estado. El procedimiento se repite para pequeños valores sucesivos de T hasta que exista un flujo óptimo finito.

El resultado de este cálculo es una curva de proyección de costo. Esta curva es lineal y convexa, ya que v aumenta conforme T disminuye, y sabemos que la representación gráfica de flujo mínimo contra v tiene estas características. La pendiente negativa en el punto T es igual al costo marginal de disminuir la duración del proyecto en una unidad de tiempo, y esperamos que este costo marginal aumente conforme T disminuye. La siguiente gráfica (fig 4.1.3) es una representación típica de una curva de proyección de costo.

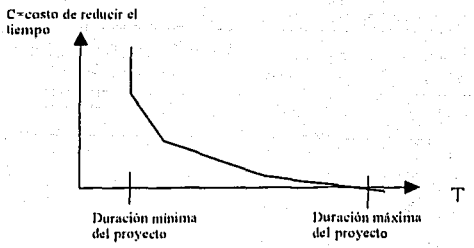


Fig 4.1.3.

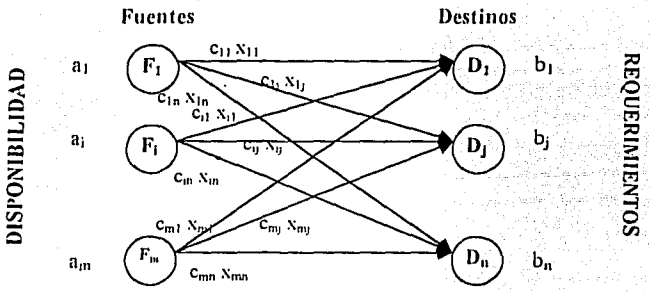
La interpretación física de las variables u_i, tiempo en el que suceden los eventos, y t_{ij}, duración de la tarea, es obvia. Sin embargo, no es tan simple interpretar a x_{ij}. La variable x_{ij} representa la cantidad de dinero que estamos dispuestos a gastar para reducir t_{ij} en una unidad de tiempo. Si $0 < x_{ij} < c_{ij}$, no estamos dispuestos a gastar exactamente lo necesario para reducir t_{ij}. Y si $x_{ij} > c_{ij}$, estaremos dispuestos a gastar una cantidad mayor a c_{ij} para reducir t_{ij}, pero es imposible reducir t_{ij} más allá de b_{ij} (tiempo de urgencia).

4.2 EL PROBLEMA DE TRANSPORTE Y TRANSBORDO

Tal vez uno de los problemas más importantes de programación lineal es el método de transporte el que se interesa en la distribución de cualquier artículo desde cualquier grupo de centros de suministro, llamado fuente, hacia cualquier grupo de centros receptores, llamados destinos, de modo que se minimicen los costos totales de distribución. En el modelo de transporte todos los coeficientes de las variables en las restricciones son iguales a uno, o sea:

$$a_{ij} = 1, \text{ para todo i y j.}$$

Si representamos gráficamente:



Donde

x_{ij} = es la unidad a enviar desde la fuente i-ésima (i=1,...,m) al destino j-ésimo (j=1,...,n)
c_{ij} = es el costo de enviar una unidad desde el fuente i-ésima (i =1,...,m) al destino j-ésimo (j =1,...,n)
a_i = es la disponibilidad (oferta) en las unidades de la fuente i-ésima (i=1,...,m)
b_j = es el requerimiento (demanda) en unidades del destino j-ésimo (j=1,...,n)

En el problema de transporte es muy importante que lo disponible sea igual a lo requerido, oferta sea igual a la demanda. Una manera de formularlo en términos generales es:

$$\text{Minimizar } Z = \sum_{i=1}^{m} c_{ij} \sum_{j=1}^{n} x_{ij}$$

sujeto a:

$$\sum_{i=1}^{m} x_{ij} = b_i \qquad i = 1,\ldots,m$$

$$\sum_{j=1}^{n} x_{ij} = a_j \qquad j = 1,\ldots,n$$

$$x_{ij} \geq 0 \text{ para todo i y j}$$

Hay que notar la siguiente relación:

$$\left.\begin{array}{l} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{m} a_{ij} \\ \\ \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{m} b_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \end{array}\right\} \begin{array}{l} \text{disponible = requerido} \\ \\ \text{oferta = demanda} \end{array}$$

Cuando un problema de transporte tiene todavía los puntos intermedios en los cuales no se dispone ni se requiere de los bienes, entonces el problema de transporte se convierte en un problema de transbordo. Por lo tanto un problema lineal de transbordo se puede tratar como un problema de transporte, o sea con los mismos métodos, pero el algoritmo de flujo con costo mínimo resuelve problema de transbordo directamente.

El problema de transbordo es una forma del problema de flujo a costo mínimo en el cual para cada vértice i hay un numero b_i dado, y en vez de la ley de conservación ordinaria se requiere que:

$$\sum_{j} x_{ji} - \sum_{j} x_{ij} \geq b_i$$

si

$b_i < 0$ el vértice es de oferta
$b_i = 0$ el vértice i es de transbordo
$b_i > 0$ el vértice i es de demanda,

Cada arco (i, j) tiene asignado un costo de flujo a_{ij}, y se asume que las capacidades de los arcos son infinitas. Si este no es el caso, entonces se dice que es un problema de transbordo con capacidades.

El problema de transporte de Hitchcock – Koopmans es un problema de transbordo con una gráfica bipartida G = (S, T, A) en la cual todos sus vértices de oferta se encuentran en S, todos sus vértices de demanda en T, y todos los arcos estan dirigidos de S a T, a demás los vértices de transbordo (o sea, los nodos intermedios) se eliminan. Otro método particular que vamos a ver más adelante en este capítulo es el problema de asignación. Es el caso especial del problema de transporte en el cual el número de vértices de oferta es igual al número de vértices de demanda y cada $b_i = \pm 1$.

Los números a_{ij}, c_{ij}, b_i y b_j se representan en la gráfica G como se muestra en la figura 4.2.1:

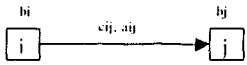


Fig 4.2.1

Es evidente que el problema de transbordo se puede reducir al problema de flujo a costo mínimo convencional con un vértice fuente s y un vértice destino t. Hay que hacer notar que si el problema tiene una solución factible, entonces la suma de las ofertas no debe ser menor a la suma de las demandas. Esto es:

$$-\sum_{b_i<0} b_i \geq \sum_{b_i>0} b_i = v$$

Tomando por hecho que el costo de cualquier ruta dirigida de un vértice oferta a un vértice demanda no es negativo, de modo que exista una solución óptima en la cual las restricciones de las demandas se satisfacen con una igualdad vamos a asociar al vértice fuente s un arco (s, i) hacia cada vértice de oferta i con capacidad $c_{si} = -b_i$ y costo $a_{si} = 0$, y al vértice destino t un arco (j, t) desde cada vértice de demanda j con capacidad $c_{jt} = b_j$ y costo $a_{jt} = 0$, reestableciendo la ley de conservación en todos los vértices. Así entonces, un flujo a costo mínimo de valor v da una solución al problema de trasbordo. Si alguna ruta dirigida de un vértice de oferta a un vértice de demanda tiene costo negativo, será necesario introducir cotas inferiores a los arcos (j, t).

Queda evidente que el problema de flujo a costo mínimo es un problema de transbordo con capacidades para un valor de flujo v deseado, donde b_s y b_t se toman de siguiente manera:

$$b_s = -v \text{ y } b_t = v.$$

Lo que es realmente sorprendente es que el problema de transbordo con capacidades y por consiguiente el problema de flujo a costo mínimo, pueden ser reducidas al problema de transporte de Hitchcock-Koopmans con capacidades.

Hay que notar que, es posible asumir que todas las ofertas y demandas en el problema de transbordo deben ser satisfechas con igualdad estricta i.e., la suma de las demandas debe ser igual a la suma de las ofertas. Si esto no es así, introducimos un vértice de demanda adicional con arcos dirigidos de los vértices de oferta a él. Y a cada arco de estos le asignamos una capacidad lo suficientemente grande y costo cero.

Esto nos da un problema de la forma:

$$\text{Minimizar } a_{ij}x_{ij}$$

$$\text{sujeto a : } \sum_j x_{ij} - \sum_j x_{ji} = b_i$$
$$0 \le x_{ij} \le c_{ij}$$

donde:

$b_i < 0$	si	i es un vértice oferta
$b_i = 0$	si	i es un vértice transbordo
$b_i > 0$	si	i es un vértice demanda

Ahora creamos una nueva red $\bar{G}$ de 2n vértices, en la cual cada vértice i de la red G de transbordo es representada por 2 vértices i y i', y un arco (i, i') con capacidad $\bar{c}_{ii'} = \infty$, y costo $\bar{a}_{ii'} = 0$. Para cada arco (i, j) de G se crea un arco (i, j') en $\bar{G}$ con capacidad $\bar{c}_{ij'} = c_{ij}$ y costo $\bar{a}_{ij'} = a_{ij}$. A los vértices de la nueva gráfica les asignamos valores $\bar{b}_i$, tal que el valor absoluto de cada $\bar{b}_i$ sea lo suficientemente granda, y $\bar{b}_i + \bar{b}_{i'} = b_i$.

Vamos a modificar las restricciones de capacidad de la red $\bar{G}$ subdividiendo cada arco (i, j) de $\bar{G}$ en 3 arcos (i, k), (k', k) y (k', j), donde k y k' son vértices nuevos introducidos por la subdivisión. Hay que notar que el vértice k tiene grado de salida cero y el vértice k' grado de entrada cero.

Los números en los vértices $\hat{b}_i$ tendrán la siguiente forma:

$\hat{b}_i = b_i$
$\hat{b}_k = \bar{c}_{ij}$
$\hat{b}_{k'} = -\bar{c}_{ij}$
$\hat{b}_j = b_j$

El costo $\hat{a}_{ik} = \bar{a}_{ij}$ y todos los demás arcos con costo cero. La capacidad de todos los arcos es infinita. La eliminación de las capacidades de los arcos se muestra a continuación:

El arco (i, j) de la red $\bar{G}$ se puede observar en la figura siguiente:

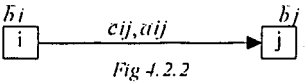


Fig 4.2.2

En la red Ĝ se transforma en:

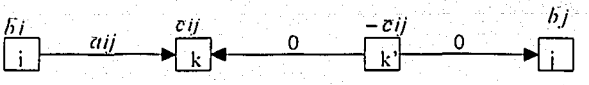


fig 4.2.3

Si $\bar{G}$ es bipartida con n vértices y m arcos, entonces $\hat{G}$ es bipartida con n+2m vértices y 3m arcos.

Ejemplo.- Si se tiene el problema con la función objetivo siguiente:

Minimizar $5x_{12} + 3x_{13} + 4x_{23} + 20x_{24} + 12x_{34} + 10x_{35} + 9x_{54} + 10x_{46} + 12x_{56}$

sujeto a:

$$-x_{12} - x_{13} = -5$$
$$x_{12} - x_{23} - x_{24} = -3 \qquad 0 \le x_{23} \le 6$$
$$x_{13} + x_{23} - x_{35} - x_{34} = 0 \qquad 0 \le x_{24} < \infty$$
$$x_{24} + x_{34} + x_{54} - x_{46} = 0 \qquad 0 \le x_{34} \le 1$$
$$x_{35} - x_{54} - x_{56} = 2 \qquad 0 \le x_{35} < \infty$$
$$x_{56} + x_{46} = 6 \qquad 0 \le x_{54} \le 9$$
$$0 \le x_{12} \le 4 \qquad 0 \le x_{46} \le 6$$
$$0 \le x_{13} \le 7 \qquad 0 \le x_{56} \le 2$$

La red de transbordo G de este problema esta representado en la siguiente figura:

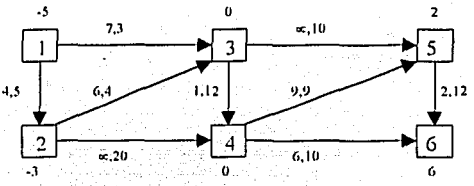
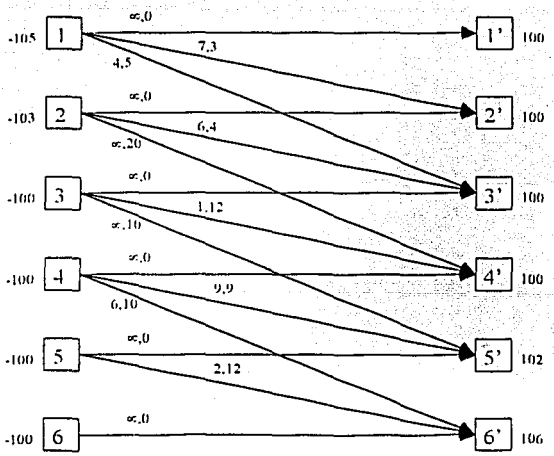


Fig 4.2.4

Y ahora veremos la red de transporte $\bar{G}$ equivalente a la red original G:



Al eliminar las capacidades de $\bar{G}$ se obtiene la siguiente forma de $\hat{G}$ (fig 4.2.5):

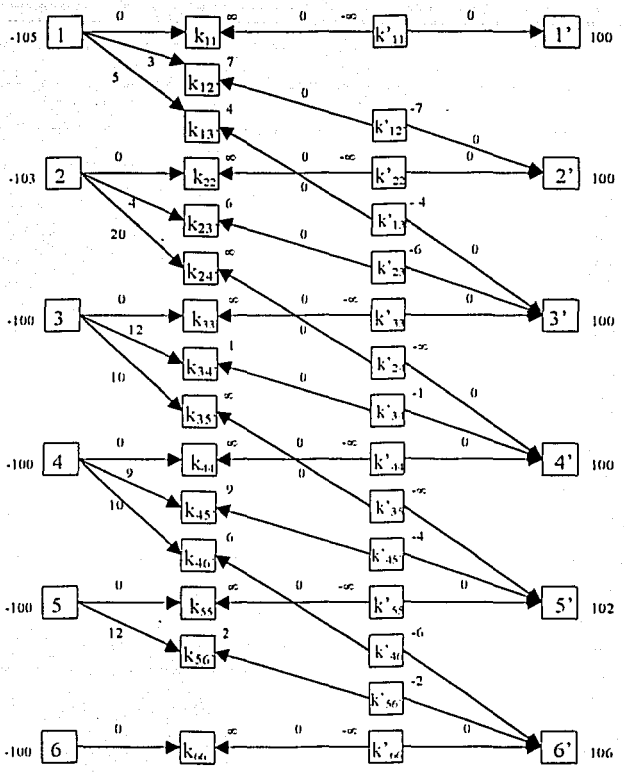


Fig 4.2.5

Pero la pregunta es: ¿cómo aplicar el método de transición de estado al problema de transporte de Hitchcock – Koopmans? Eso lo vamos a ver enseguida.

El problema de transporte, con o sin restricciones, se convierte fácilmente en un problema de flujo, introduciendo un vértice fuente s con arcos (s, i) a cada vértice oferta i con capacidad $c_{si} = -b_i$ y costo $a_{si} = 0$; igual que el vértice destino t con arcos (j, t) desde cada vértice demanda j con capacidades $c_{jt} = b_j$ y costo $a_{jt} = 0$, y un arco de regreso (t, s) con costo $a_{ts} = 0$ y capacidades $l_{ts} = c_{ts} = v$, donde v es igual a la suma de las ofertas que a su vez es igual a la suma de las demandas.

Para esta red de flujo, las soluciones primal x = (0) y dual u = (0) son soluciones factibles y sólo el arco (t, s) esta en mal estado, y el número de buen estado es v. Conforme el cálculo del algoritmo de transición de estado se efectúa, el arco (t, s) es el único arco que siempre esta en mal estado, tal como ocurrió en el problema de planeación de proyectos.

Ejemplo. - Una compañía "X" dispone de 2 fábricas. Una se encuentra ubicada en la ciudad A y otra en la ciudad B, y tiene 3 clientes, uno en K, otro en L y tercero en M. Los costos de transportar la mercancía de las fábricas a los almacenes de los clientes, así como las ofertas y las demandas se muestran en la tabla 7.

	K	L	M	Oferta
A	250	170	180	350
B	250	180	140	600
Demanda	325	300	275	

Tabla 7

Gráficamente planteamiento de este problema como el problema de flujo a costo mínimo, será:

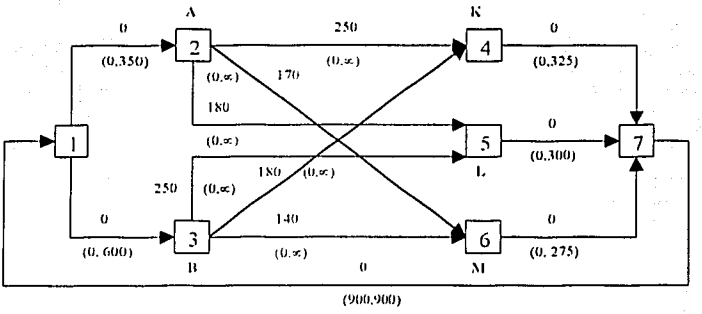


Fig 4.2.6

Todos los procedimientos computacionales para resolver el problema de transporte virtualmente pueden ser interpretados como adaptaciones, variaciones o especializaciones del método de transición de estado cuando se aplica de esta manera.

4.3 EL PROBLEMA DE ASIGNACIÓN

El problema de asignación es el tipo especial de problemas lineales con la estructura de transporte, cuando m = n y en el que los recursos se asignan a las actividades en términos de uno a uno, o sea, la oferta en cada origen es de valor uno y la demanda en cada destino es también de valor uno. Así entonces, cada recurso o cesionario (puede ser un empleado, una máquina o un tiempo determinado) debe asignarse de modo único a una actividad particular o asignación (que son: tarea, sitio o evento). Se tiene un costo c_{ij} asociado con el cesionario i (i=1,2.....,n) que lleva a cabo la asignación j (j=1,2,....,n) y el objetivo consiste en asignar a cada hombre un trabajo de modo que el costo total sea mínimo.

La formulación de un problema de asignación es la siguiente:

$$\text{Minimizar } z = \sum_{i=1}^{n} \sum_{j=1} c_{ij} x_{ij}$$

sujeto a:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,...,m$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1,...,n$$

$$x_{ij} \geq 0 \qquad i = 1,...,m \quad j = 1,...,n$$

Las variables x_{ij} sólo pueden tomar el valor 0 ó 1. Es 1 si el origen i se hace corresponder al destino j, y es valor 0 en caso contrario.

A los problemas de asignación se les pueden aplicarse los algoritmos de transporte por ser éste su caso especial, pero debido a la estructura propia de los problemas de asignación, existen métodos específicos para obtener la solución y son llamados *algoritmos de asignación* los cuales son más eficientes que el método simplex o el método de transporte para tratar este tipo de problemas ya que las restricciones en problemas de asignación, admiten exactamente m variables positivas en cada solución básica factible. Eso quiere decir que a cada persona sólo se le asigna un trabajo y viceversa. La condición necesaria y suficiente para que el problema tenga solución, es que estén balanceados, i.e., la oferta total tiene que ser igual a la demanda total. En el caso de que no exista este balance, primero habrá que balancearlo de la misma manera que a un problema de transporte.

Suponiendo que se tiene n hombres y n trabajos. El costo de asignar al hombre i al trabajo j es a_{ij}.

Para encontrar la solución al problema anterior se construye una gráfica bipartida con n vértices en cada una de sus partes (condición que puede garantizarse creando hombres o trabajos ficticios), y al arco (i, j) se le asocia el costo a_{ij} y la capacidad máxima infinita $c_{ij} = \infty$. También se le añade un vértice fuente s con un arco (s, i) a cada vértice de la primera parte, y un vértice destino t con arco (j, t) de cada vértice de la segunda parte. Se toman $c_{si} = 1$, $a_{si} = 0$ para toda i, y $c_{jt} = 1$, $a_{jt} = 0$ para toda j. Un flujo entero de valor v (v es el mínimo de los hombres o trabajos que se tiene originalmente, antes de introducir vértices ficticios) con costo mínimo produce una solución al

problema. Este problema se transforma en uno de flujo a costo mínimo de la misma manera que el problema de flujo con costo mínimo que vimos en el capítulo dos. Y la red asociada al problema de asignación es de la figura 4.3.1:

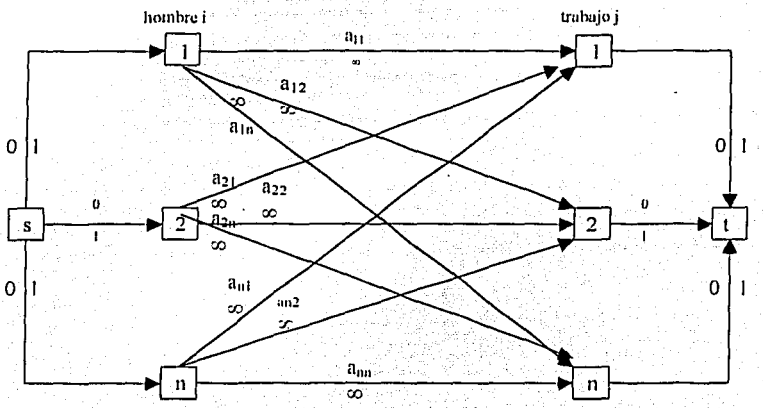


Fig 4.3.1

Y la red asociada al problema de asignación transformado en un problema de flujo a costo mínimo es la siguiente:

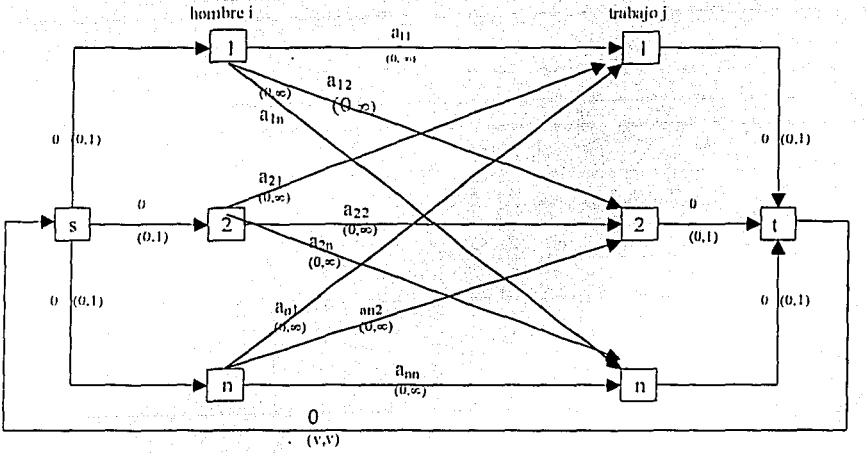


Fig 4.3.3

Ahora veremos un problema real de asignación y cómo lo podemos transformar o convertir en uno de flujo a costo mínimo.

Ejemplo. - Una competencia de relevos de trescientos metros incluye a cuatro nadadores, quienes nadan sucesivamente cien metros de dorso, pecho y mariposa. Un entrenador tiene cuatro nadadores muy veloces cuyos tiempos esperados en los eventos individuales en segundos estan dados en la tabla 8:

Nadador	Dorso	Pecho	Mariposa
1	65	73	63
2	67	70	65
3	68	72	69
4	67	75	70

Tabla 8

Lo que nos interesa para poder resolver este problema es: cómo debe el entrenador asignar a los nadadores en los diferentes eventos de la competencia a fin de minimizar la suma de sus tiempos.

El planteamiento de este problema como un problema de flujo a costo mínimo es la siguiente:

$$\text{Minimizar } 65x_{11} + 67x_{21} + 68x_{31} + 67x_{41} + 73x_{12} + 70x_{22} + 72x_{32} + 75x_{42} + 63x_{13} + 65x_{23} + 69x_{33} + 70x_{43}$$

Sujeto a:

$$x_{ts} - (x_{s1} + x_{s2} + x_{s3} + x_{s4}) = 0$$
$$x_{s1} - (x_{11} + x_{12} + x_{13} + x_{14}) = 0$$
$$x_{s2} - (x_{21} + x_{22} + x_{23} + x_{24}) = 0$$
$$x_{s3} - (x_{31} + x_{32} + x_{33} + x_{34}) = 0$$
$$x_{s4} - (x_{41} + x_{42} + x_{43} + x_{44}) = 0$$
$$(x_{11} + x_{21} + x_{31} + x_{41}) - x_{1t} = 0$$
$$(x_{12} + x_{22} + x_{32} + x_{42}) - x_{2t} = 0$$
$$(x_{13} + x_{23} + x_{33} + x_{34}) - x_{3t} = 0$$
$$(x_{14} + x_{24} + x_{34} + x_{44}) - x_{4t} = 0$$

$$0 \le x_{s1} \le 1 \qquad 0 \le x_{1t} \le 1$$
$$0 \le x_{s2} \le 1 \qquad 0 \le x_{2t} \le 1$$
$$0 \le x_{s3} \le 1 \qquad 0 \le x_{3t} \le 1$$
$$0 \le x_{24} \le 1 \qquad 0 \le x_{4t} \le 1$$
$$0 \le x_{11} \le \infty \qquad 0 \le x_{12} \le \infty$$
$$0 \le x_{13} \le \infty \qquad 0 \le x_{14} \le \infty$$
$$0 \le x_{21} \le \infty \qquad 0 \le x_{22} \le \infty$$
$$0 \le x_{23} \le \infty \qquad 0 \le x_{24} \le \infty$$
$$0 \le x_{31} \le \infty \qquad 0 \le x_{32} \le \infty$$
$$0 \le x_{33} \le \infty \qquad 0 \le x_{34} \le \infty$$
$$0 \le x_{41} \le \infty \qquad 0 \le x_{42} \le \infty$$
$$0 \le x_{43} \le \infty \qquad 0 \le x_{44} \le \infty$$
$$v = 3 \le x_{ts} \le 3 = v$$

CONCLUSIONES

En el trabajo presente se han visto y analizado los problemas de flujo de redes, tratando de presentar todas las ideas relacionadas de una manera unificada y de acuerdo con los objetivos establecidos.

Los problemas de flujo en redes a costo mínimo se pueden resolver por medio del método simplex, sin embargo lo que se busca es una simplificación que pueda aplicarse directamente a la red, sin la necesidad de las tablas de simplex. Vimos que cualquier problema de flujo con costo mínimo lo podemos convertir en un problema de programación lineal de flujo a costo mínimo y resolverlo a través del algoritmo de transición de estado, que es un algoritmo de tipo primal-dual para el problema de flujo a costo mínimo. El método es mucho más sencillo y eficiente que la técnica de simplex y sus variantes. Requiere que las capacidades, sean números enteros para todos los arcos y que la red a resolver sea circular. Dicho método resulta fácil de aplicar para resolver éste tipo de problemas debido a que tiene como ventajas que no aumenta el tamaño de la red, como ya lo mencionamos no se usan las tablas de simplex, los cálculos resultan ser más sencillos, y el número de iteraciones es menor. Por eso es una de las técnicas más aplicadas en diversos campos de la ingeniería.

Además el trabajo se basa en explicar en qué consisten los problemas de flujo máximo, de ruta más corta, de flujos con restricciones, de transporte y transbordo, de asignación, y en cómo pueden ser resueltos a través del método de transición de estado. Los modelos de redes que se vieron en el trabajo han sido ejemplificadas para mostrar su aplicabilidad y la manera de pasar del marco teórico al practico.

Cabe mencionar que las posibles desventajas del método es la construcción de subgráficas después de cada iteración, además que, estas subgráficas tienen que elaborarse con mucho cuidado, ya que al determinar un arco en buen estado, este puede ser eliminado de la subgráfica, así reduciendo el tamaño de la red y el trabajo. Pero el inconveniente está en que no se puede eliminar un arco de la subgráfica, aunque este en buen estado, si al quitarlo, no se puede encontrar un circuito para mejorar el estado de algún otro arco que este en mal estado. Eso quiere decir que hay que ser cuidadoso en la eliminación de los arcos, porque al equivocarse para obtener la solución del problema tendríamos que realizar más iteraciones de lo necesario.

Las posibles aplicaciones más importantes están en campos tan diversos como la investigación de operaciones, ingeniería eléctrica y optimización combinatoria, la planeación, la ingeniería, la administración, la química, la economía, entre otros. Y son tan importantes y útiles debido a que muchísimas situaciones de la vida diaria, relaciones interpersonales, circulación vial, rutas de aviación etc. pueden ser formuladas como los modelos matemáticos, además de que son de fácil comprensión para personas que tienen poco conocimiento en investigación de operaciones y por su facilidad de representarlas gráficamente. Además los modelos de redes de flujo, igual que otros modelos de investigación de operaciones, sirven de auxiliares en el proceso de la toma de decisiones.

Este método debería de ser visto en Teoría de Optimización II ya que da un enfoque realista, diferente y es una opción más a utilizar en la soluciones de problemas de investigación de operaciones. Es fácil de comprender y aplicar, y por ser una técnica diferente de lo acostumbrado, ya que es el método suplementario del método simplex.

APÉNDICE

Un problema de programación lineal es un problema de minimizar o maximizar la función lineal sujeta a las restricciones lineales que pueden ser desigualdad, igualdad o ambas.

El desarrollo de programación lineal se sitúa en la segunda mitad del siglo XX. Su impacto desde 1950 fue extraordinario, y hasta la fecha actual es una herramienta muy útil que se ha ido extendiendo muy rápidamente.

La programación lineal trata los problemas de asignar recursos limitados entre las actividades competidoras de la mejor forma posible usando un modelo matemático para describir el problema que se tenga. Y la expresión de "programación lineal" significa la planificación de actividades para poder obtener un resultado óptima, o sea, para que el resultado alcance la meta especifica en la mejor forma posible. Para poder representar un problema de optimización como un problema de programación lineal deben de existir los elementos nesesarios y deben de ser conocidas y estimados.

El modelo primal de un el problema de programación lineal tiene siguiente forma:

$$\text{Minimizar } z = cx$$

$$\begin{aligned} &\text{sujeto a:} && A x^1 \geq b^1 \\ & && A x^2 = b^2 \\ &\text{con} && x^1 \geq 0,\ x^2 \text{ no restringida, y } x \in R^n \end{aligned}$$

Las matrices respectivas del modelo son siguientes:

$$A = \begin{matrix} \left[\begin{matrix} A^1 \\ A^2 \end{matrix}\right] \\ n \end{matrix} \begin{matrix} \} \text{p} \\ \} \text{m-p} \end{matrix} \quad b = \begin{matrix} \left[\begin{matrix} b^1 \\ b^2 \end{matrix}\right] \\ 1 \end{matrix} \begin{matrix} \} \text{p} \\ \} \text{m-p} \end{matrix} \quad x = \begin{matrix} \left[\begin{matrix} x^1 \\ x^2 \end{matrix}\right] \\ 1 \quad 1 \end{matrix} \begin{matrix} \} \text{q} \\ \} \text{n-q} \end{matrix}$$

Ahora, si tomemos el siguiente sistema de desigualdades:

$$A x = b$$

Seleccionando el conjunto de m columnas linealmente independientes del arreglo A, donde B es la matriz de m × m, determinada por las columnas, entonces, la matriz B es singular y la ecuación

$$B x_B = b$$

tiene una solución únicamente para el m-vector x_B. Para obtener una solución para $Ax = \bar{b}$ tenemos que hacer que se la expresión: $x = (x_B, 0)$ sea cierta, donde x_B es la solución básica de $Ax = \bar{b}$ respecto a la base B de A. Y los componentes de x asociadas a las columnas de B se son las variables básicas. Si una o más variables básicas de la solución básica es igual a cero, entonces esta solución es la *solución básica degenerada*.

Para que un vector sea factible las siguientes restricciones tienen que ser ciertas:

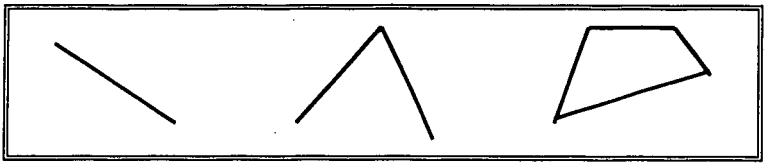
$$Ax = \bar{b}$$
$$x \geq 0$$

Una solución básica factible es aquella que es factible para estas restricciones. Y si ademas es la solucion degenerada, entonces la solucion al problema se llamará solución basica factible degenerada.

Como vimos en el capítulo uno, un politopo conexo es un conjunto que puede ser expresado como un número finito de semiespacios cerrados. Hay que notar que los politopos conexos son varios conjuntos que se obtienen como las soluciones a un conjunto de desigualdades de forma:

$$a_1 x^T \leq b_1$$
$$a_2 x^T \leq b_2$$
$$\vdots$$
$$a_m x^T \leq b_m.$$

donde cada desigualdad define un semiespacio y el grupo de soluciones es la intersección de estos semiespacios. Los politopos pueden ser vacío, acotado o no acotado. En la siguiente figura se puede ver ejemplos de los politopos:



Un punto x de un conjunto convexo C se llama *extremo* de C, si no existen dos puntos distintos x_1 y x_2 en C, tales que $x = \alpha x_1 + (1 - \alpha)x_2$ para algún α con $0 < \alpha < 1$. Por lo tanto, un punto extremo es el que se encuentra exactamente dentro del segmento de recta que está uniendo dos puntos del conjunto. Por ejemplo: los puntos extremos de un triangulo son sus vértices.

Teorema: Teorema fundamental de programación lineal

Dado un problema de programación lineal:

Minimizar $c^T x$

sujeto a: $A x = b$
$x \geq 0$,

donde A es la matriz de m × n de rango m, entonces:

1. Si existe una solución factible $\Rightarrow$ existe una solución factible básica.
2. Si existe una solución factible óptima $\Rightarrow$ existe una solución factible óptima básica.

Supongamos que K es el politopo conexo para todos los vectores x que satisface las restricciones:

$$A x = b$$
$$x \geq 0$$

Entonces, el vector x es un punto extremo de K, si y sólo si x es una solución básica factible para estas restricciones.

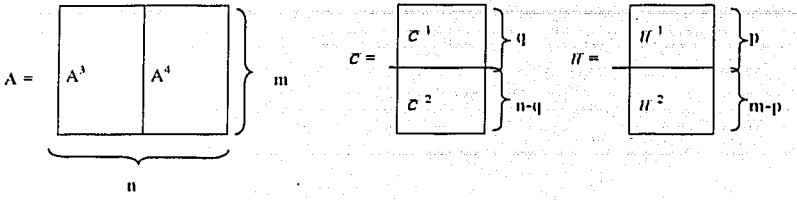
La equivalencia entre puntos extremos y soluciones básicas factibles demuestra algunas propiedades geométricas de politopo conexo K que define el conjunto de dichas restricciones en un problema de programación lineal. Entre las propiedades mencionadas anteriormente están:

1. Si el conjunto conexo K correspondiente a $A x = b, x \geq 0$ es no vacío, entonces se tiene al menos un punto extremo.
2. Si existe para un problema de programación lineal una solucion óptima finita, entonces, hay una solucion óptima finita que es un punto extremo del conjunto de restricciones.
3. El conjunto de restricciones K que corresponde a $A x = b, x \geq 0$ tiene a lo más un número finito de puntos extremos.
4. Si el politopo conexo K correspondiente a $A x = b, x \geq 0$ es acotado, entonsec K es un politopo conexo, o sea, K se compone de puntos que son combinaciones de un número finito de puntos.

Como vimos en el capitulo uno dentro de las definiciones, todo el problema de programación tendrá un problema correspondiente en su forma dual que se define así:

Maximizar $w = b^T \pi$
Sujeto a: $(A^3)^t \pi \leq c^1$
$(A^4)^t \pi \leq c^2$
$\pi^1 \geq 0$, π^2 no restringida, donde $\pi \in \mathbb{R}^m$

y las matrices respectivas son:



La existencia del problema dual al problema de programación lineal es la consecuencia directa del espacio vectorial adjunto que esta siempre asociado a una transformación lineal sobre un espacio vectorial.

Ejemplo: Suponiendo que se tiene el siguiente problema en su forma primal:

$$\text{Minimizar } z = 5x_1 + 2x_2 + x_3$$

Sujeto a:
$$\begin{aligned} &2x_1 + 3x_2 + x_3 \geq 20 \\ &6x_1 + 8x_2 + 5x_3 \geq 30 \\ &7x_1 + x_2 + 3x_3 = 40 \\ &x_1 + 2x_2 + 4x_3 = 50 \\ &x_1, x_2 \geq 0 \\ &x_3 \text{ no restringida.} \end{aligned}$$

Donde las matrices y vectores están particiónados como se muestra:

$$A = \left[\begin{array}{ccc} 2 & 3 & 1 \\ 6 & 8 & 5 \\ \hline 7 & 1 & 3 \\ 1 & 2 & 4 \end{array}\right] \qquad b = \left[\begin{array}{c} 20 \\ 30 \\ \hline 40 \\ 50 \end{array}\right] \qquad x = \left[\begin{array}{c} x_1 \\ x_2 \\ \hline x_3 \end{array}\right]$$

Por lo tanto, como vimos anteriormenta podemos determinar que m = 4, n = 3, p = 2 y q = 2.

Pasando dicho problema primal a su respectiva forma dual, éste tendrá siguiente forma:

$$\text{Maximizar } w = 20u_1 + 30\,u_2 + 40\,u_3 + 50\,u_4$$

Sujeto a:
$$\begin{aligned} &2u_1 + 6u_2 + 7u_3 + u_4 \leq 5 \\ &3u_1 + 8u_2 + u_3 + 2u_4 \leq 2 \\ &u_1 + 5u_2 + 3u_3 + 4u_4 \leq 1 \\ &u_1, u_2 \geq 0 \\ &u_3, u_4 \text{ no restringidas} \end{aligned}$$

La partición que van a tener ahora las matrices es la siguiente:

$$A = \left[\begin{array}{cc|c} 2 & 3 & 1 \\ 6 & 8 & 5 \\ \hline 7 & 1 & 3 \\ 1 & 2 & 4 \end{array}\right] \qquad c = \left[\begin{array}{c} 5 \\ 2 \\ \hline 1 \end{array}\right] \qquad u = \left[\begin{array}{c} u_1 \\ u_2 \\ \hline u_3 \\ u_4 \end{array}\right]$$

Teorema. – Teorema de ortogonalidad de soluciones óptimas

Si x y u son soluciones factibles a:

Primal	Dual
Minimizar $c\,x$	Maximizar $u\,b$
sujeto a: $A\,x \geq b$	sujeto a: $u\,A \leq c$
$x \geq 0$	$u \geq 0$,

entonces

x y u son óptimas si y sólo si, $(u\,A - c)\,x = u\,(A\,x - b) = 0$

Esto es cierto en caso de que para j = 1, 2, ...,n, $x_j > 0$ y entonces

$$\sum_{i=1}^{m} u_i\, a_{ij} = c_j$$

y para i = 1, 2, ..., m con $u_i > 0$ implica que

$$\sum_{j=1}^{m} a_{ij}\, x_j = b_i$$

por lo tanto $c\,x = u\,b$.

Queda demostrada la relación simétrica del problema dual con el problema primal de un problema de programación lineal.

Ahora veremos el teorema de Hoffman y Kruskal *ref. (14)* que es de suma importancia y el cual nos asegura la solución entera al problema de flujo en redes.

Teorema.- Teorema de Hoffman y Kruskal

En un problema de programación lineal con restricciones $A\,x = b$, $x \geq 0$, donde A es una matriz de componentes enteras con renglones linealmente independientes (rango (A) = m), y b es un vector de componentes enteras. Entonces, las 3 condiciones siguientes son equivalentes:

d) El determinante de cualquier base B es ± 1.
e) Los puntos extremos del politopo convexo C definido por $A\,x = b$, $x \geq 0$ son enteros, para todo vector b de componentes enteras.
f) La matriz inversa B^{-1} de cualquier base B sólo tiene componentes enteras.

Demostración:

b) $\Rightarrow$ b)
Sea $x = (x^B, x^R)$ un punto extremo del politopo convexo C y B su base asociada. Por la regla de Cramer tenemos que el det(B) = $\pm$ 1 implica que B^{-1} sólo tiene componentes enteras. Entonces, si b es un vector de enteros $x^B = B^{-1}$ b es un vector de enteros.

a) $\Rightarrow$ c)
Sea B una base y sea y cualquier vector de componentes enteras, tal que, $y + B^{-1} e_i \geq 0$. (donde e_i es la i-ésima componente unitaria del vector columna). Sea $z = y + B^{-1} e_i \geq 0$, de aquí que $Bz = By + BB^{-1} e_i = By + e_i$ es un vector de componentes enteras ya que B, y y e_i son de componentes enteras. Como b puede ser cualquier vector de componentes enteras tomemos $b = Bz$. Ahora $Bz = b$ y $z \geq 0$ que muestra que z es un punto extremo del politopo C definido por $A\,x = b$, $x \geq 0$; Por b) z es un vector de componentes enteras, pero $z - y = B^{-1} e_i$ de donde se sigue que $B^{-1} e_i$ es de componentes enteras. El vector $B^{-1} e_i$ es el i-ésimo vector columna de la matriz B^{-1}; este argumento se puede repetir para i = 1, 2, ..., m y se demuestra que B^{-1} es una matriz de enteros.

b) $\Rightarrow$ a)
Sea B una base de A. Asumimos que B es una matriz de componentes enteras y que el det(B) es entero. Por la condición c), B^{-1} es una matriz de componentes enteras, entonces, el det (B^{-1}) también es entero. Pero, $\det(B)\det(b^{-1}) = \det(BB^{-1}) = \det(I_m) = 1$, lo que implica que $\det(B) = \det(B^{-1}) = \pm 1$.

Corolario 1.- Sea C' el politopo convexo definido por las restricciones de desigualdad $A'x \leq \bar{b}$, y $x \geq o$, donde A' es una matriz de componentes enteras. Las 3 condiciones siguientes son equivalentes:

a) A' es totalmente unimodular.
b) Todos los puntos extremos de C' son enteros para cualquier vector $\bar{b}'$ de componentes enteros.
c) Toda submatriz de A' no singular tiene una matriz inversa de componentes enteras.

Demostración.- La demostración de esta corolario se basa en demostrar la equivalencia de a) con a'), b) con b') y c) con c').

a)⇔ a')
Sea A = (A', I). Si M es cualquier submatriz de A' de rango m-k, entonces, una base de A puede ser encontrada permutando renglones, de la forma:

$$B = \left[\frac{M}{N}\,\frac{0}{Ik}\right]$$

Donde Ik es la matriz identidad de k×k. Entonces, como det(B) = det(M) el det(B) = ± 1.

b) ⇔ b')
Sea $x \in R^n$ una solución básica factible de

$$A'x' \leq b'$$
$$x \geq 0$$

Donde $\bar{b} \in Z^m$. Por lo tanto, para toda $y \in \mathbb{R}^m$ con $y \geq 0$, el vector (x, y) satisface:

$$(A, I).\left[\frac{x}{y}\right]' = .(\bar{b}).$$

Pero por el teorema, $(x, y) \in \mathbb{Z}^{n+m}$, por lo tanto $x \in \mathbb{Z}^n$.

c) ⇔ c')
Tenemos la siguiente hipótesis: $B^{-1} \in \Lambda(\mathbb{Z})$. Y queremos demostrar que toda submatriz M invertida de A' es tal que $M^{-1} \in \Lambda(\mathbb{Z})$.

Similarmente a lo que se hizo en a) $\Leftrightarrow$ a') tenemos que:

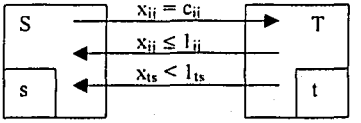
$$B = \begin{bmatrix} M & 0 \\ N & Ik \end{bmatrix} \qquad \text{y} \qquad B^{-1} = \begin{bmatrix} M^{-1} & 0 \\ S & Ik \end{bmatrix}$$

Pero, por la hipótesis tenemos que $B^{-1} \in \Lambda(\mathbb{Z})$. Por lo tanto, $M^{-1} \in \Lambda(\mathbb{Z})$.

El siguiente teorema que veremos fue desarrollado por Hoffman (1960) *ref. (6)* y es el que nos asegura una solución factible en un problema de flujo con capacidades.

> *Teorema.*- En una red con funciones de capacidad mínima y máxima, una circulación factible existe sí, y solo sí:
>
> $$\sum_{i \in T, j \in S} l_{ij} \leq \sum_{i \in S, j \in T} c_{ij}$$
>
> para todos los enlaces – (S, T).

Demostración.- Supongamos que una ruta aumentante de flujo no puede ser encontrada. Sea (t, s) el arco para el cual no se puede encontrar la ruta aumentante de flujo con $x_{t,s} < l_{ts}$. Sea S el conjunto de vértices que pueden ser alcanzados desde s por una ruta aumentante de flujo, y T es el conjunto de vértices que no pueden ser alcanzados desde s. Para cada arco (i, j) dirigido de T a S $x_{ij} \leq l_{ij}$. Como se ve en la siguiente figura:



El flujo neto a través del enlace – (S, T) es cero, i.e.,

$$\sum_{i \in S, j \in T} x_{ij} = \sum_{i \in T, j \in S} x_{ij}$$

pero

$$\sum_{i \in S, j \in T} x_{ij} = \sum_{i \in S, j \in T} c_{ij}$$

y

$$\sum_{i \in T, j \in S} x_{ij} \leq \sum_{i \in T, j \in S} l_{ij}$$

Con desigualdad estricta debido al arco (t, s). Hemos construido un conjunto de corte – (S, T) para el cual:

$$\sum_{i \in S, j \in T} c_{ij} \leq \sum_{i \in T, j \in S} l_{ij}$$

ANEXO

IMPLEMENTACIÓN COMPUTACIONAL DEL MÉTODO DE TRANSICIÓN DE ESTADO

LISTADO DEL PROGRAMA DEL MÉTODO DE TRANSICIÓN DE ESTADO

```
{Programa para la transición de estados, utilizando el algoritmo
Out of Kilter, para resolver problemas de flujo a costo mínimo y
capacidad máx. Sasha Kondratenko;                    Enero 2003}


Program TransicionEstado;
Uses Crt;

Const
 Infinito = 1.7E38;

Var
  Arco, Flujo,
  EtiqPerm,
  EtiqTmp,
  ArrEnlace      : Array[1..10,1..2]of Integer;
  Costo, X, U,
  ArrPi          : Array[1..10]of Real;
  Ciclo          : Array[1..10]of Integer;
  ColorArc       : Array[1..10]of String;
  nNodos, nArcos,
  ContEtiqPerm,
  ContEtiqTmp,
  ApEtiqPerm,
  T,S,ContEnlace,
  nNodosCiclo    : Integer;
  Ch             : Char;
  {
   Arco: Vector que contiene el nodo inicial y final Arco[i,j]
   Flujo: Vector que contiene el el flujo min y max de cada arco [min,
max]
   EtiqPerm: Vector que contiene la etiqueta permanente del arco n
[arco,etiqueta]
   Costo: Vector que contiene el costo de cada arco
   ColorArc: Guarda el Color de cada arco
   X, U : Vectores que guardan el resultado del primal y dual.
   nNodos: Numero de nodos
   nArcos: Numero de Arcos
   T, S : Indica el nodo inicial T y el final S para poner etiq,
permanente S.
   ContEnlace: Contador de elementos en el enlace
   nNodosCiclo : Numero de nodos del ciclo
   Ciclo: Arreglo que contiene la ruta del flujo aumentado
  }
```

```
Procedure PasoUno; Forward;
Procedure UnoUno; Forward;

{*****************************************************************}

Procedure Leer; {Lee el numero de nodos y los arcos}
Var
 i : ShortInt;
Begin

  {Pide los elementos}
   Write('Numero de Nodos: ');
   ReadLn(nNodos);
   Write('Numero de Arcos');
   ReadLn(nArcos);
   For i := 1 to nArcos Do
   Begin
     WriteLn;
     WriteLn('--- Arco ',i,' ---');
     Write('Nodo Inicial: ');
     ReadLn(Arco[i,1]);
     Write('Nodo Final: ');
     ReadLn(Arco[i,2]);
     Write('Costo: ');
     ReadLn(Costo[i]);
     Write('Flujo Minino: ');
     ReadLn(Flujo[i,1]);
     Write('Flujo Maximo: ');
     ReadLn(Flujo[i,2]);
   End;
End;

Function Capacidades:Boolean;
{Devuelve True si cumple con la ley de conservacion (SumC(i,j) >=
SumL(j,k))
 y False en caso contrario}
Var
 i, j, k,Cont : Integer;
 SumC, SumL   : Integer; {SumC: Suma Flujo Max, SumL: Suma Flujo Min}
Begin
  Capacidades:= True;
  SumC := 0;
  SumL := 0;

  For i:= 1 to nNodos Do
  Begin
    For j:= 1 to nNodos Do
    Begin
      For Cont := 1 to nArcos Do
      Begin
        If (Arco[Cont,1] = i) And (Arco[Cont,2] = j) Then SumC:=
SumC+Flujo[Cont,2];
      End;
      For k:= 1 to nNodos Do
      Begin
```

```
          For Cont := 1 to nArcos Do
          Begin
            If (Arco[Cont,1] = j) And (Arco[Cont,2] = k) Then SumL:=
 SumL+Flujo[Cont,1];
          End;
        End;
      End;
    End;
    If SumC >= SumL Then Capacidades := True
    Else
    Begin
      Capacidades := False;
      WriteLn;
      WriteLn('No cumple con la ley de Conservacion');
      WriteLn;
    End;
 End;  {Procedure}

 Function NumEstado:Real;
 {Regresa el numero de estado de cada arco u la suma de estos}
 Var
   i         : Integer;
   SumK,Du,
   sum       : Real;

 Begin
   SumK := 0;
   Sum := 0;
   For i := 1 to nArcos Do
   Begin
     If ColorArc[i] = 'Amarillo*' Then Du := (U[Arco[i,1]] - U[Arco[i,2]])
     Else Du := (U[Arco[i,2]] - U[Arco[i,1]]);

     If Du = Costo[i] Then
     Begin
       If (X[i] >= Flujo[i,1]) And (X[i] <= Flujo[i,2]) Then Sum := 0
       Else
       Begin
         If X[i] < Flujo[i,1] Then Sum := Flujo[i,1] - X[i];
         If X[i] > Flujo[i,2] Then Sum := X[i] - Flujo[i,2];
       End;
     End
     Else
     Begin
       If Du < Costo[i] Then Sum := Abs(X[i] - Flujo[i,1]);
       If Du > Costo[i] Then Sum := Abs(X[i] - Flujo[i,2]);
     End;
     SumK := SumK + Sum;
   End;
   NumEstado := SumK;
 End;

 Function CostodeFlujo: Real;
 {Calcula el costo del flujo de la solucion Primal}
 Var
   i     : Integer;
   Kosto : Real;
```

```
Begin;
  Kosto := 0;
  For i := 1 to nArcos Do
  Begin
    Kosto := Kosto + (X[i]*Costo[i]);
  End;
  CostodeFlujo := Kosto;
End;

Procedure InvertirDir(Indice: Integer);
Var
  Tmp : Integer;
Begin
  If ColorArc[Indice] = 'Amarillo*' Then
  Begin
    Tmp := Arco[Indice,1];
    Arco[Indice,1] := Arco[Indice,2];        {Invierte la direccion}
    Arco[Indice,2] := Tmp;
  End;
End;

Procedure ColorearArcos;
{ Colorea todos los arcos para utilizar el teorema de Minty}
Var
   i  : Integer;
   Du : Real;

Begin
  For i := 1 to nArcos Do
  Begin
    InvertirDir(i);
    Du := U[Arco[i,2]] - U[Arco[i,1]];

    If ((X[i] > Flujo[i,1] ) And ((X[i] < Flujo[i,2]) And (Du =
Costo[i]))) Then
        ColorArc[i] := 'Verde';
    If (X[i] < Flujo[i,1]) And (Du = Costo[i]) Then ColorArc[i] :=
'Amarillo';
    If (X[i] < Flujo[i,1]) And (Du < Costo[i]) Then ColorArc[i] :=
'Amarillo';
    If (X[i] > Flujo[i,1]) And (Du < Costo[i]) Then ColorArc[i] :=
'Amarillo*';
    If (X[i] = Flujo[i,1]) And (Du < Costo[i]) Then ColorArc[i] :=
'Rojo';
    If (X[i] < Flujo[i,2]) And (Du > Costo[i]) Then ColorArc[i] :=
'Amarillo';
    If (X[i] > Flujo[i,2]) And (Du > Costo[i]) Then ColorArc[i] :=
'Amarillo*';
    If (X[i] >= Flujo[i,2]) And (Du = Costo[i]) Then ColorArc[i] :=
'Amarillo*';
    If (X[i] < Flujo[i,2]) And (Du > Costo[i]) Then ColorArc[i] :=
'Rojo';

    InvertirDir(i);
    Du := U[Arco[i,2]] - U[Arco[i,1]];
  End;
```

```
End;


Procedure BuscArcoMalEstado;
{Busca un arco en mal estado y le da etiqueta permanente 0}
Var
  Terminar : Boolean;
  Cont     : Integer;
  Du       : Real;

Begin
  Terminar := False;
  Cont := 1;
  Repeat
    Du := U[Arco[Cont,2]] - U[Arco[Cont,1]];
    If ColorArc[Cont] = 'Amarillo' Then
    Begin
      If (X[Cont] <> Flujo[Cont,1]) And (Du <> Costo[Cont]) Then
      Begin
        T:= Arco[Cont,1];
        S:= Arco[Cont,2];
        Terminar := True;
      End;
    End;
    If ColorArc[Cont] = 'Amarillo*' Then
    Begin
      If (X[Cont] <> Flujo[Cont,2]) And (Du <> -Costo[Cont]) Then
      Begin
        T:= Arco[Cont,1];
        S:= Arco[Cont,2];
        Terminar := True;
      End;
    End;
    Inc(Cont);
  Until (Terminar = True) Or (Cont = nArcos);
  WriteLn('Arco para convertir a buen estado: (T= ',T,'    S= ',S,')');
  WriteLn;
  EtiqPerm[1,1] := S;
  EtiqPerm[1,2] := 0;
  Inc(ContEtiqPerm);
  ApEtiqPerm := 0;
  WriteLn('Etiqueta Permanente: ',S,'  ',0);
End;

Function ArcosenBuenEstado:Integer;
{Regresa el numero de arcos en buen estado}
Var
 i, Cont : Integer;
Begin
  Cont := 0;
  For i := 1 to nArcos Do
  Begin
    If ColorArc[i] = 'Verde' Then Inc(Cont);
    If ColorArc[i] = 'Amarillo' Then
    Begin
      If (X[i] = Flujo[i,1]) And ((U[Arco[i,2]] - U[Arco[i,1]]) =
Costo[i]) Then Inc(Cont);
```

```
    End;
    If ColorArc[i] = 'Amarillo*' Then
    Begin
      If (X[i] = Flujo[i,2]) And ((U[Arco[i,2]] - U[Arco[i,1]]) = -
Costo[i]) Then Inc(Cont);
    End;
    If ColorArc[i] = 'Rojo' Then Inc(Cont);
  End;
  ArcosenBuenEstado := Cont;
End;

Procedure EtiqAmarilloVerde;
Var
 bVar,
 bCierto : Boolean;
 i, j, k : Integer;
Begin
  For i := 1 to nArcos Do
  Begin
    If ColorArc[i] <> 'Rojo' Then
    Begin
      If Arco[i,1] = EtiqPerm[ApEtiqPerm,1] Then
      Begin
        bVar := True;
        For j := 1 to ContEtiqPerm Do
        Begin
           If bVar = True Then
              If EtiqPerm[j,1] = Arco[i,2] Then bVar := False;
        End;
        If bVar = True Then
        Begin
          Inc(ContEtiqPerm);
          EtiqPerm[ContEtiqPerm,1] := Arco[i,2];
          EtiqPerm[ContEtiqPerm,2] := Arco[i,1];
          bVar := True;
          For j := 1 to ContEtiqTmp Do
          Begin
            If bVar = True Then
            Begin
              If EtiqTmp[j,1] = Arco[i,2] Then
              Begin
                Dec(ContEtiqTmp);
                EtiqPerm[ContEtiqPerm,2] := EtiqTmp[j,2];
                For k:= j to ContEtiqTmp Do
                Begin
                  EtiqTmp[k,1]:= EtiqTmp[K+1,1];
                  EtiqTmp[k,2]:= EtiqTmp[K+1,2];
                End;
                bVar := False;
              End;
            End;
          End;
        End;
      End;

      If Arco[i,2] = EtiqPerm[ApEtiqPerm,1] Then
      Begin
```

```
        If ColorArc[i] = 'Verde' Then
        Begin
          bCierto := True;
          For j := 1 to ContEtiqPerm Do
          Begin
            If bCierto = True Then
               If EtiqPerm[j,1] = Arco[i,1] Then bCierto := False;
          End;
        End;
        If bCierto = True Then
        Begin
          Inc(ContEtiqPerm);
          EtiqPerm[ContEtiqPerm,1] := Arco[i,2];
          EtiqPerm[ContEtiqPerm,2] := Arco[i,1];
          bCierto := True;
          For j := 1 to ContEtiqTmp Do
          Begin
            If bCierto = True Then
            Begin
              If EtiqTmp[j,1] = Arco[j,1] Then
              Begin
                Dec(ContEtiqTmp);
                For k:= j to ContEtiqTmp Do
                Begin
                  EtiqTmp[k,1]:= EtiqTmp[K+1,1];
                  EtiqTmp[k,2]:= EtiqTmp[K+1,2];
                End;
                bCierto := False;
              End;
            End;
          End;
          k := ContEtiqPerm-1;
          For j := 1 to k Do
          Begin
            If EtiqPerm[j,1] = EtiqPerm[ContEtiqPerm,1] Then
Dec(ContEtiqPerm);
          End;
        End;
      End;
    End;
  End;
End;

Procedure EtiquetarRojo;
{ Da etiqueta permanete a un arco rojo
  si cumple con las condiciiones señaladas
}
Var
 bVar : Boolean;
 i,j,k : Integer;
Begin
  For i := 1 to nArcos Do
  Begin
    If  Arco[i,1] = EtiqPerm[ApEtiqPerm,1] Then
    Begin
      If ColorArc[i] = 'Rojo' Then
      Begin
```

```
        If (X[i] = Flujo[i,1]) And ((Costo[i]-U[Arco[i,2]]) <
(ArrPi[Arco[i,2]])) Then
        Begin
          bVar := True;
          For j := 1 to ContEtiqPerm Do
          Begin
            If bVar = True Then
            Begin
              If EtiqPerm[j,1] = Arco[i,2] Then bVar := False;
            End;
            If bVar = True Then
            Begin
              bVar := True;
              For j := 1 to ContEtiqTmp Do
              Begin
                If bVar = True Then
                Begin
                  If EtiqTmp[j,1] = Arco[i,2] Then
                  Begin
                    Dec(ContEtiqTmp);
                    For k:= j to ContEtiqTmp Do
                    Begin
                      EtiqTmp[k,1] := EtiqTmp[k+1,1];
                      EtiqTmp[k,2] := EtiqTmp[k+1,2];
                    End;
                    bVar := False;
                  End;
                End;
                Inc(ContEtiqTmp);
                EtiqTmp[ContEtiqTmp,1] := Arco[i,2];
                EtiqTmp[ContEtiqTmp,2] := Arco[i,1];
              End;
            End;
          End;
        End;
      End;
    End;
    If Arco[i,2] = EtiqPerm[ApEtiqPerm,1] Then
    Begin
      If ColorArc[i] = 'Rojo' Then
      Begin
        If (X[i] = Flujo[i,2]) And ((U[Arco[i,1]]-U[Arco[i,1]] -
Costo[i]) < (ArrPi[Arco[i,2]])) Then
        Begin
          bVar := True;
          For j := 1 to ContEtiqTmp Do
          Begin
            If bVar = True Then
              If EtiqPerm[j,1] = Arco[i,1] Then bVar := False;
          End;
          If bVar = True Then
          Begin
            bVar := True;
            For j := 1 to ContEtiqTmp Do
            Begin
              If EtiqTmp[j,1] = Arco[i,1] Then
              Begin
```

```
                    Dec(ContEtiqTmp);
                    For k:= j to ContEtiqTmp Do
                    Begin
                      EtiqTmp[k,1]:= EtiqTmp[k+1,1];
                      EtiqTmp[k,2]:= EtiqTmp[k+1,2];
                    End;
                    bVar := False;
                  End;
                End;
                Inc(ContEtiqTmp);
                EtiqTmp[ContEtiqTmp,1] := Arco[i,1];
                EtiqTmp[ContEtiqTmp,2] := Arco[i,2];
                ArrPi[Arco[i,1]] := (U[Arco[i,2]] - U[Arco[i,1]])-Costo[i];
              End;
            End;
          End;
        End;
      End;
    End;

Procedure Enlace;
{Encuentra el enlace entre los arco T,S}
Var
  i, j, k : Integer;
  bVar    : Boolean;
Begin
  ContEnlace := 0;
  For i := 1 to nArcos Do
  Begin
    For j := 1 to ContEtiqPerm Do
    Begin
      If Arco[i,1] = EtiqPerm[j,1] Then
      Begin
        bVar := True;
        For k := 1 to  ContEtiqPerm Do
        Begin
          If Arco[i,2] = EtiqPerm[k,1] Then bvar := False;
          If (bVar = True) And (ColorArc[i] <> 'Verde') Then
          Begin
            Inc(ContEnlace);
            ArrEnlace[ContEnlace,1] := Arco[i,1];
            ArrEnlace[ContEnlace,2] := Arco[i,2];
          End;
        End;
      End;
      If Arco[i,2] = EtiqPerm[j,1] Then
      Begin
        bVar := True;
        For k := 1 to ContEtiqPerm Do
        Begin
          If Arco[i,1] = EtiqPerm[k,1] Then bVar := False;
          If (bVar = True) And (ColorArc[i] <> 'Verde') Then
          Begin
            Inc(ContEnlace);
            ArrEnlace[ContEnlace,1] := Arco[i,1];
            ArrEnlace[ContEnlace,2] := Arco[i,2];
          End;
```

```
        End;
      End;
    End;
  End;
  WriteLn('Enlace: ');
  For i:= 1 to ContEnlace Do
  Begin
    WriteLn(ArrEnlace[i,1], '      ',ArrEnlace[i,2]);
  End;
End;


Function CalcularEpsilon: Real;
{Calcula todas la epsilon del enlace y obtiene la menor de estas}
Var
 i,j        : Integer;
 Ep, Epsilon: Real;
Begin
  Epsilon := Infinito;
  For i := 1 to nArcos Do
  Begin
    For j := 1 to ContEnlace Do
    Begin
      If (Arco[i,1] =ArrEnlace[j,1]) And (Arco[i,2] = ArrEnlace[j,2])
Then
      Begin
        Ep := Infinito;
        If ColorArc[i] = 'Amarillo' Then
        Begin
          If (Flujo[i,1] <= X[i]) And ((U[Arco[i,2]]-U[Arco[i,1]]) >
Costo[i]) Then
            Ep := (U[Arco[i,2]] - U[Arco[i,1]]) - Costo[i]
        End;
        If ColorArc[i] = 'Amarillo*' Then
        Begin
          If (Flujo[i,2] <= X[i]) And ((U[Arco[i,1]]-U[Arco[i,2]]) <
Costo[i]) Then
            Ep := (Costo[i] - U[Arco[i,1]]) + U[Arco[i,2]]
        End;
        If ColorArc[i] = 'Rojo' Then
        Begin
          Ep := Abs(Costo[i] - (U[Arco[i,2]]+U[Arco[i,1]]));
        End;
        If Epsilon >  Ep Then Epsilon := Ep;
      End;
    End;
  End;
  CalcularEpsilon := Epsilon;
End;

Procedure CambioNumNodos;
{ Realiza los procesos del paso 3}
Var
 epsilon    : Real;
 i,j,k,l,m,
 cBuenEst,
 ContbEst   : Integer;
```

```
 bVar,Boole: Boolean;
Begin
  WriteLn;
  WriteLn('Paso 3');
  WriteLn;
  cBuenEst:= ArcosenBuenEstado;
  Enlace;
  Epsilon := CalcularEpsilon;
  WriteLn;
  WriteLn('Epsilon = ',Epsilon:0:4);
  If Epsilon < Infinito Then
  Begin
    For i := 1 to nNodos Do
    Begin
      bVar := True;
      j := 1;
      Repeat
         If EtiqPerm[j,1] = j Then bVar := False;
         Inc(j);
      Until (bVar = False) Or (j > ContEtiqPerm);
      If bVar = True Then U[i] := U[i]+Epsilon;
    End;
    WriteLn;
    WriteLn('Solucion Dual: ');
    For j := 1 to nNodos Do
    Begin
      WriteLn('U(',j,')= ',U[j]:0:4);
      ContBEst := ArcosenBuenEstado;
      If ContBEst < cBuenEst Then   {Caso 2}
      Begin
        WriteLn('Caso 2');
        PasoUno;
      End
      Else
      Begin
        For k := 1 to nNodos Do
        Begin
          bVar := True;
          j := 1;
          Repeat
             If EtiqPerm[j,1] = k Then bVar := False;
             Inc(j);
          Until (bVar = False) Or (j > ContEtiqPerm);
          If bVar = True Then
          Begin
            ArrPi[k] := ArrPi[k]-Epsilon;
            If ArrPi[k] = 0 Then
            Begin
              WriteLn('Caso 3');
              Boole := True;
              For l := 1 to ContEtiqPerm Do
              Begin
                If Boole = True Then
                Begin
                  If EtiqPerm[l,1] = k Then Boole := False;
                End;
              End;
```

```
                If Boole = True Then
                Begin
                  Inc(ContEtiqPerm);
                  EtiqPerm[ContEtiqPerm,1] := k;
                  bVar := True;
                  For m := 1 to ContEtiqTmp Do
                  Begin
                    If bVar = True Then
                    Begin
                      If EtiqTmp[m,1] = k Then
                      Begin
                        EtiqPerm[ContEtiqPerm,2]:= EtiqTmp[m,2];
                        Dec(ContEtiqTmp);
                        For j:= m to ContEtiqTmp Do
                        Begin
                          EtiqTmp[j,1] := EtiqTmp[j+1,1];
                          EtiqTmp[j,1] := EtiqTmp[j+1,1];
                        End;
                        bVar := False;
                      End;
                    End;
                  End;
                End;
              End;
            End;
          End;
        End;
        WriteLn;
        For k:= 1 to nNodos Do
        Begin
          WriteLn;
          WriteLn('Pi [',K,']=',ArrPi[K]:0:4);
        End;
        WriteLn;
        WriteLn('Etiquetas Permanentes');
        For k:= 1 to ContEtiqPerm Do
        Begin
          WriteLn;
          WriteLn(EtiqPerm[k,1], '      ', EtiqPerm[k,2]);
        End;
        ColorearArcos;
        ApEtiqPerm := 0;
        UnoUno;
      End;
    End
    Else
    Begin
      WriteLn('-------------------------------------------------------');
      WriteLn('No existe flujo factible');
    End;
    Ch := ReadKey;
End;


Procedure EncontrarCiclo;
Var
  i,j : Integer;
```

```
Begin
  Ciclo[1] := EtiqPerm[1,1];
  Ciclo[2] := EtiqPerm[ContEtiqPerm,1];
  Ciclo[3] := EtiqPerm[ContEtiqPerm,2];
  j := 3;
  Repeat
    For i := ContEtiqPerm-1 Downto 1 Do
    Begin
      If EtiqPerm[i,1] = Ciclo[j] Then
      Begin
        Inc(j);
        Ciclo[j] := EtiqPerm[i,2];
      End;
    End;
  Until Ciclo[1] <> Ciclo[j];
  nNodosCiclo := J-1;
  WriteLn('Ciclo Amarillo Verde:');
  For i := 1 to nNodosCiclo Do
  Begin
    Write(Ciclo[i],';');
  End;
  WriteLn;
  Ch := ReadKey;
End;

Function CalcularDelta: Real;
{Realiza el calculo de todas las deltas y regresa la menor}
Var
 i,j      : Integer;
 d, Delta : Real;
 bVar     : Boolean;
Begin
  EncontrarCiclo;
  Delta := Infinito;
  For i := 1 to nArcos Do
  Begin
    For j := 1 to nNodosCiclo-1 Do
    Begin
      If (Arco[i,1] = Ciclo[j+1]) And (Arco[i,2] = Ciclo[j]) Then
      Begin
        d := Infinito;
        If (U[Arco[i,2]] - U[Arco[i,1]]) = Costo[i] Then
        Begin
          If (ColorArc[i] = 'Verde') Or (ColorArc[i] = 'Amarillo') Then d
:= Flujo[i,2]-X[i];
          If (ColorArc[i] = 'Verde') Or (ColorArc[i] = 'Amarillo*') Then
d := X[i] - Flujo[i,1];
          If Delta > d Then Delta := d;
        End
        Else
        Begin
          If (ColorArc[i] = 'Amarillo') Or (ColorArc[i] = 'Amarillo*')
Then d:= Abs(X[i]-Flujo[i,1]);
        End;
        If Delta > d Then Delta := d;
      End;
    End;
```

```
  End;
  CalcularDelta := Delta;
End;


Procedure CambiodeCirculacion;
{Realiza los procesos del Paso 2}
Var
 Delta        : Real;
 i,j,k, Du    : Integer;
 bVar, Increm: Boolean;
Begin
  WriteLn;
  WRiteLn('Paso 2');
  Delta:= CalcularDelta;
  WriteLn('Delta = ',Delta:0:4);
  If Delta < Infinito Then
  Begin
    For i := 1 to nNodosCiclo Do
    Begin
      bVar := True;
      j := 1;
      Repeat
        If (Arco[j,1] =Ciclo[i+1]) And (Arco[j,2] = Ciclo[i]) Then
        Begin
          If ColorArc[j] = 'Verde' Then
          Begin
            If Increm = True Then
            Begin
              X[j] := X[j]+Delta;
              Increm := True;
            End
            Else
            Begin
              X[j] := X[j]-Delta;
              Increm := False;
            End;
          End;
          If ColorArc[j] = 'Amarillo' Then
          Begin
            X[j] := X[j]+Delta;
            Increm := True;
          End;
          If ColorArc[j] = 'Amarillo*' Then
          Begin
            X[j] := X[j]-Delta;
            Increm := False;
          End;
          bVar := False;
        End;
        Inc(j);
      Until (bVar = False) Or (j > nArcos);
    End;
    WriteLn;
    WriteLn('Solucion Primal');
    For k := 1 to nArcos Do
    Begin
```

```
      If ColorArc[j] = 'Amarillo*' Then
         WriteLn('X(',Arco[k,2],',',Arco[k,1],')=',X[k]:0:5)
      Else WriteLn('X(',Arco[k,1],',',Arco[k,2],')=',X[k]:0:5)
    End;
    PasoUno;
  End
  Else
  Begin
    WriteLn('-------------------------------------------------------');
    WriteLn('No existe una solucion optima que sea finita');
    Ch := ReadKey;
  End;
End;

(*************************************************************************)

Procedure UnoDos;
{Verifica si el nodo T tiene etiqueta permanente y salta al paso 2 o 1.1
segun la condicion}
Var
 bVar : Boolean;
 Cont : Integer;

Begin
  WriteLn;
  WriteLn('Paso 1.2');
  bVar := True;
  Cont := 1;
  Repeat
    If EtiqPerm[Cont,1] = T Then bVar := False;
    Inc(Cont);
  Until (bVar = False) Or (Cont > ContEtiqPerm);
  If bVar = True Then UnoUno Else CambiodeCirculacion;
End;

Procedure UnoUno;
Var
  i : Integer;
Begin
  WriteLn;
  WriteLn('Paso 1.1');
  Inc(ApEtiqPerm);
  If ApEtiqPerm <= ContEtiqPerm Then
  Begin
    EtiqAmarilloVerde;
    EtiquetarRojo;
    WriteLn('Etiquetas Permanentes');
    For i := 1 to ContEtiqPerm Do WriteLn(EtiqPerm[i,1],'
',EtiqPerm[i,2]);
    WriteLn;
    WriteLn('Etiquetas Temporales');
    For i := 1 to ContEtiqTmp Do WriteLn(EtiqTmp[i,1],'
',EtiqTmp[i,2]);
    WriteLn;
    For i:= 1 to nNodos Do
    Begin
      WriteLn('Pi [',i,']= ',ArrPi[i]:0:4);
```

```
      End;
      UnoDos;
    End
    Else
    Begin
      Dec(ApEtiqPerm);
      CambioNumNodos;
    End;
End;

Procedure PasoUno;
Var
 i              : Integer;
 NumEst, nCosto : Real;
Begin
  WriteLn('Paso 1');
  nCosto := CostodeFlujo;
  NumEst := NumEstado;
  WriteLn('Numero de Estado: ',NumEst:0:4);
  WriteLn('Costo de solucion primal: ',nCosto:0:4);
  If NumEst <> 0 Then
  Begin
    ColorearArcos;
    WriteLn;
    For i := 1 to nArcos Do
    Begin
      If ColorArc[i] = 'Amarillo*' Then Write('Color del
Arco(',Arco[i,2],',',Arco[i,1],') es: ')
      Else Write('Arco(',Arco[i,1],',',Arco[i,2],') = ');
      WriteLn(ColorArc[i]);
    End;
    For i := 1 to nArcos Do
    Begin
      ArrPi[i] := Infinito;
      WriteLn('Pi[',i,']= ', ArrPi[i]:0:6);
    End;
    ContEtiqPerm := 0;
    ContEtiqTmp := 0;
    BuscArcoMalEstado;
    UnoUno;
  End
  Else
  Begin
    WriteLn('------------------------------------------------------------');
    WriteLn('Costo Optimo: ',nCosto:0:4);
    WriteLn('Solucion Optima; ');
    WriteLn('Flujo Optimo; ');
    For i := 1 to nArcos Do
    Begin
      If ColorArc[i] = 'Amarillo*' Then
WriteLn('X(',Arco[i,2],',',Arco[i,1],')= ',X[i]:0:4)
      Else WriteLn('X(',Arco[i,1],',',Arco[i,2],')= ',X[i]:0:4);
    End;
  End;
  WriteLn;
  Ch := ReadKey;
  WriteLn('Solucion Optima Dual');
```

```
  WriteLn('Nodos Optimos');
  For i := 1 to nArcos Do
  Begin
    WriteLn('U(',i,')= ',U[i]:0:4);
  End;
  WriteLn('------------------------------------------------------------');
End;

Begin
  {Inicializa los arreglos}
  Fillchar(Arco,SizeOf(Arco),0);
  Fillchar(Flujo,SizeOf(Flujo),0);
  Fillchar(Costo,SizeOf(Costo),0);
  Fillchar(EtiqPerm,SizeOf(EtiqPerm),1);
  Fillchar(EtiqTmp,SizeOf(EtiqTmp),1);
  Fillchar(ColorArc,SizeOf(ColorArc),' ');
  Fillchar(ArrPi,SizeOf(ArrPi),0);
  Fillchar(Ciclo,SizeOf(Ciclo),0);
  Fillchar(X,SizeOf(X),0);
  Fillchar(U,SizeOf(U),0);
  Fillchar(ArrEnlace,SizeOf(ArrEnlace),0);

  Leer;
  If (Capacidades = True) Then PasoUno;
End.
```

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

1. Hiller y Leiberman "Introducción a Investigación de Opercaones", ed. McGraw-Hill
2. Lawler E. L. "Combinatorial Optimization Networks and Matroides", Holt, Reinhart and Wiston, N.Y., 1976
3. Goldberg A.V. and Tarjan R. E. "Finding Minimum-Cost Circulations by Successive Approximation" In Mathematics of Operations Research, vol.15, No.3, August 1990, pp.430-466
4. "Graph Theory "Harary, ed. Addison Wesley
5. "Graphs as Mathematical Models", Gary Chartrand, ed. Prindle Webwr &Schmidt Incorporated
6. Wilson Robin "Introducción la teoría de grafos". ed. Alianza Universidad
7. Bazaraa, M. S., Jarvis, J. J., "Programación lineal y flujo en redes". Limusa, 1981
8. Luenberger, D.E.: "Programación lineal y no lineal". Addison-Wesley, 1989
9. Taha, H.A. "Investigación de Operaciones". Una Introducción. (1998). Pearson. Prentice Hall

BIBLIOGRAFÍA COMPLEMENTARIA

1. WINSTON, Wayne L., "Investigación de Operaciones: Aplicaciones y Algoritmos". Grupo Editorial Iberoamericana, S.A. de C.V., 1994
2. BRONSON R. "Investigación de Operaciones". Teoría y 310 problemas resueltos. Serie Schaum. McGraw-Hill
3. J. Prawda. " Métodos y Modelos de Investigación de Operaciones" vol I., Modelos Deterministicos
4. "Graph Theory"., Cristofides Academic Press
5. SARABIA VIEJO A. "Problemas de Investigación Operativa". Ediciones ICAI
6. Ma. Del Carmen Hernández Ayuso. "Introducción a la teoría de redes", ed., Sociedad Matemática Mexicana. textos 12. nivel medio

7. Lara Rosano Felipe., "Análisis topoógico de la confiabilidad de redes de flujo sujetos a restricciones de calidad". Tesis de Posgrado, División de estudios de Posgrado, Facultad de Ingeniería UNAM

8. Mayra Olguin Rosas., "Teoría de redes y sus aplicaciones". Tesis de licenciatura, Matemáticas Aplicadas y Computación, UNAM., 1991

9. Moreno Moreno Martha., "Transporte eléctrico en México, análisis de eficiencia y la calidad". Tesis de maestria, Facultad de Ingeniería, UNAM., 1999

10. Cadena Landeta Alberto., "Algoritmos para resolver problemas de redes". Tesis de posgrado, Facultad de Ingenieria, UNAM., 1982

PÁGINAS DE INTERNET

1. http://www.lafacu.com/apuntes/economia/Modelo_de_transporte_final/default.htm
2. http://www.geocities.com/SiliconValley/Pines/7894/modelos/transporte.html
3. http://www.geocities.com/leoprimus/SimplexHelp5.htm
4. http://www.info-ab.uclm.es/scc-ab/plan/mycopera.html
5. http://ingenet.ulpgc.es/~ablesa/optimizacion/introoptinetred.htm
6. http://w3.mor.itesm.mx/~optimiza/opti9901/capa2/ruta+corta.html
7. http://www.tecnun.es/asignaturas/io1/pagina_2.html
8. http://delta.cs.cinvestav.mx/~schapa/proyectos/ecdtf/reporte4/node4.html
9. http://150.214.55.100/curso9899/204b3/Pmo/Traspal/pm6.htm
10. http://www.google.com/search?q=Edmonds+y+Karp&hl=es&lr=lang_es&ie=UTF-8&start=10&sa=N
11. http://www.esi2.us.es/~dco/docencia.htm

REFERENCIAS

1. Hitchcock F. L., "The Distribution of a Product from Several Sources to Numerous Localities". Journal of Mathematics and Physics. Vol. 20, No. 2, April 1941., p. 224.

2. Koopmans T. C., "Optimum Utilization of the Transportation System". Proceedings of the International Statistical Conferences. Washington, D. C., 1947. Vol. 5, 1949. p. 136 (Also in Scientific Papers of Tjalling C. Koopmans. Sprilinger-Verlag. New York, 1970, p. 184).

3. Dantzig G. B., "Application of the Simplex Method to a Transportation Problem". In Acticity Analysis of Production and Allocation. T. C. Koopmans. ed., John Wiley and Sons, New York. 1951.

4. Minty G. J., "Solving Steady-State Nonlinear Network of "Monotone2 Elements", I. R. E. Trans. Circuit. Theory, CT-8, 1961, p. 99-104.

5. http://www.esi2.us.es/~dco/docencia.htm

6. Hoffman A. J., "Some Recent Applications of the Theory of Linear Inequalities to Extremal Combinatorial Analysis". Proc. Symposium on Appl. Math., 10, 1960.

7. Hillier, F.S.; Lieberman, G.J.: "Introducción a la investigación de operaciones". McGraw-Hill, 1990.

8. Ma. Del Carmen Hernández Ayuso, "Introducción a la teoría de redes", ed., Sociedad Matemática Mexicana, textos 12, nivel medio.

9. Clasen, R. J. "The Numerical Solution of Network Problems Using Out-of-Kilter Algorithm". RAND Report RM-5456 PR, marzo 1968.

10. Jewell, W. S., "Complex: A Complementary Slackness, Out-of-Kilter Algorithm for Linear Programming, ORC 67-6, Universidad de California, Berkeley, Calif., 1967.

11. Minty G. J., "Monotone Networks", Proc. Roy. Soc. London, Ser. A, 257, 1960., p. 194-212.

12. Fulkerson D. R., "An Out-of Kilter Method for Minimal Cost Flow Problem". Siam J. Appl. Math., 9, 1961., p. 18-27.

13. Yakovleva M. A., "Problem of Minimum Transportation Expense", in Applications of Mathematics to Economics Research, ed. By V. S. Nemchinov, Moscuw, 1959., p. 390-399.

14. Hoffman A. J Kruskal J. B., "Integral Boundary Pointe of Convex Polyhedra", in H. W. Kuhn and A. W. Turker. Linear Inequalities and Related Systems. Annals of Mathematics Study No. 38, Princeton Univ. Press, Princenton, New Jersey, 1956, p. 133-146.

15. Edmonds J. and Karp R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", J. Assoc. Comput. Math. 19, 1972. p 248-264.

16. Edmonds J. and Karp R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", J. Assoc. Comput. Math. 19, 1972. p 248-264.

17. Röck H., "Sacaling Techniques for Minmal Cost Network Flows". In U. Pape, Discrete Structures and Algorithms, Carl Hansen, Munich, 1980. p 181-191.

18. Orlin, J. B., "Genuinely Polynominal Simplez and Non-Simplex Algorithms for the Minimum Cost Flow Problem". Thechnical Report No. 11615-48, Sloan School of management, MIT, December 1984.

19. Fujishige S., "A Capacity-Rounding Algorithm for the Minimum-Cost Circulation Problem: a Dual Framework of the Tardos Algorithm", math. Programming 35, 1986, p. 298-308.

20. Galil Z. and Tardos E., "Minimum Cost flow Algorithm". J. Assoc. Comput. Math. 35, 1988, p. 374-386.

21. Orlin, J. B., "A Faster Strongly Polinomial Minimum Cost Flow Algorithm". In Proc. 20 ACM Sysp. Theory of Computing, 1988., p. 377-387.

22. Tardos E., "A Strongly Polinomial Minimum Cost Circulation Algorithm". Combinatorica 53, 1985, p. 247-255.

23. Bland R. G. and Jensen D. L. "On the Computational Behaviour of a Polynomial-Time Network Flow Algoritjm". Technical Report 661, School of Operations Research and Industrial Engineering, Cornell University, 1985.

24. Goldberg A. V. and Tarjan R. E. "Finding Minimum-Cost Circulations by Canceling negative Cycles". In Proc. 20 ACM Symp. Theory of Computing, 1988, p. 873-886.